VR Cinematic Camera

Treball Final de Grau

Grau en Disseny i Desenvolupament de Videojocs

Alumne: Rifà Planellas, Pere

Pla: 2014

Directora: Zuñiga Zarate, Ana Gabriel

Índex

Resum

En la indústria dels videojocs la utilització de cutscenes és una eina narrativa recurrent que permet als desenvolupadors presentar escenes, perills, obstacles o simplement donar informació al jugador de la situació en la que es troba. El problema resideix en que la producció d'aquestes cinemàtiques sol ser un procés costós, llarg i que sovint requereix d'una persona especialitzada per tal d'obtenir el resultat desitjat.

Amb l'objectiu de solventar aquest problema, es proposa el desenvolupament d'un prototip d'eina senzill, ràpid d'utiltzar i amb resultats de qualitat acceptable. El prototip desenvolupat fa ús de la realitat virtual i adapta funcionalitats destacades de Cinemachine, eina de referencia dins de Unity. Seguint un desenvolupament centrat en les funcionalitats, es crea un prototip que permet gravar i editar el recorregut de les càmeres en un escenari 3D per reproduir-lo a posteriori en un videojoc 3D.

Si ve el resultat final no deixa de ser un prototip, les funcionalitats analitzades en l'etapa d'investigació han estat incorporades amb èxit. Així, a pesar de no suposar una solució real al problema plantejat, hi ha la possibilitat de continuar el seu desenvolupament fins el punt de ser-ho.

Paraules clau

Camera, Cinemàtiques, Cutscenes, Eina, Unity3D, Videojocs, Realitat Virtual, Bézier

Índex de taules

Índex de figures

Glossari

CutScene: escena no interactiva que es reprodueix en un videojoc, s'utilitza com a suport narratiu.

Realitat Virtual: tecnologia immersiva que crea un món virtual amb el qual l'usuari pot interaccionar en primera persona.

Controladors HTC Vive: dispositius utilitzats per interpretar els moviments i les accions de les mans dels usuaris per interaccionar amb el software.

Ulleres de Realitat Virtual: dispositiu de visualització similar a un casc que permet reproduir imatges creades per ordinadors sobre una pantalla molt proxima a l'ull.

Unity: motor de videojocs multiplataforma utilitzat pel desenvolupament de contingut multimedia, com són els videojocs.

Timeline de Unity: eina integrada al motor de Unity utilitzada per organitzar i crear cutscenes, cinemàtiques o seqüències de gameplay.

Render: Imatge digital que es crea a partir d'un model o escenari 3D.

Corbes de Bézier: corba polinòmica creada a partir de dos o més punts i les seves tangents aplicant l'algorisme d'interpolació de Bézier.

Transformació: matriu 4x4 utilitzada en softwares 3D per representar la posició i orientació d'un objecte o d'un punt en l'espai virtual.

Gizmos: Representació gràfica dels eixos de coordenades que permet modificar la posició, rotació i escala d'una transformació.

Plugin: Aplicació que aporta una funcionalitat o característica de software extra a programa informàtic.

Frustum: Una regió de l'espai 3D delimitada per dos plans, els continguts d'aquesta regió poden aparèixer en pantalla.

Asset: és la representació d'un element que es pot utilitzar en el joc, els assets poden ser generats per Unity o externs a Unity, com models 3D o textures.

1. Introducció

1.1 Motivació

Des de fa uns anys he sentit una fascinació per la realitat virtual, els seus usos i el seu potencial.

En l'actualitat podem trobar aplicacions en diversos àmbits que utilitzen aquesta nova tecnologia per transmetre experiències immersives als usuaris, ja sigui en videojocs, en aplicacions mèdiques o en altres simulacións de tot tipus.

Al ser estudiant de disseny i desenvolupament de videojocs he volgut enfocar l'ús d'aquesta tecnologia en el procés de creació de videojocs. La idea que m'ha captat més l'atenció és la utilització de la Realitat Virtual com a eina per gravar cinemàtiques en entorns 3D per després reproduir-les en els videojocs.

Com a tal la motivació principal que m'ha portat a presentar aquest treball de fi de grau és la d'ampliar els meus coneixements de la Realitat Virtual i produir una eina que ajudi a companys desenvolupadors de videojocs en la seva feina.

1.2 Formulació del problema

En la indústria dels videojocs la utilització de cutscenes és una eina narrativa recurrent que permet als desenvolupadors presentar escenes, perills, obstacles o simplement donar informació al jugador de la situació en la que es troba.

El problema resideix en que la producció d'aquestes cinemàtiques sol ser un procés costós, llarg i que sovint requereix d'una persona especialitzada per tal d'obtenir el resultat desitjat.

Aquests trets poden suposar un inconvenient per a empreses petites que poden no disposar dels recursos necessaris, ja sigui temps, diners o personal per gravar les cutscenes.

La idea darrere aquest projecte és la de proporcionar una eina senzilla, ràpida d'utilitzar i amb resultats de qualitat acceptable per posar-la a disposició d'aquestes empreses.

L'eina s'integrarà al motor de videojocs més utilitzat, Unity, que ja disposa de solucions parcials per aquest problema.

Una d'aquestes solucions és Cinemachine, una eina professional i gratuïta proporcionada directament per l'equip de Unity. En la botiga de Unity també hi trobem el VR Cameraman que explora el mateix concepte presentat en el treball però no és compatible amb les versions recents de Unity (2018 i posteriors).

Com a tal, les eines existents no representen una solució accessible a desenvolupadors no especialitzats, la primera proporciona una eina professional on utilitzar la funcionalitat bàsica està a l'abast de molts però disposa de sistemes avançats que requereixen molta dedicació per ser utilitzats. Per altre banda VR Cameraman, a pesar de presentar una solució al problema, esta obsoleta i inoperativa per a les versions recents del motor.

Conclueix-ho que l'eina resultant d'aquest treball compartirà la idea dels creadors de VR Cameraman incorporant funcionalitats de Cinemachine i un nou disseny per tal de adaptar-se millor a la funcionalitat desitjada.

1.3 Objectius generals del TFG

L'objectiu principal d'aquest Treball de Fi de Grau es el de crear un prototip d'eina per a Unity que, amb l'ajuda de controladors HTC Vive i ulleres de Realitat Virtual, permet-hi gravar el recorregut de les càmeres, editar-lo i posteriorment reproduir-lo en una escena 3D mitjançant scripting.

Objectius:

1. Creació d'un prototip per crear i editar recorreguts de càmeres en entorns 3D per al motor Unity.
2. Implementació de la realitat virtual com a eina per controlar les càmeres i el seu recorregut en l'etapa de rodatge.
3. Implementació de funcionalitats avançades de càmera que siguin compatibles amb la Timeline de Unity.
4. Proporcionar eines de suport per millorar la usabilitat per part dels dissenyadors que utilitzin l'eina.
5. Rodatge d'una Cutscene senzilla que mostri les capacitats de l'eina.

1.4 Objectius específics del TFG

Per assolir els objectius generals definits en l'anterior apartat es proposen un seguit d'objectius intermedis que serviran com a guia de les tecnologìes i algoritmes a utilitzar. Els objectius específics seran analitzats i investigats per confirmar la necessitat d'implementar-los en el prototip d'eina, per tant poden ser acceptats o rebutjats.

Tractament de Dades:

Exportar/Importar: Serialització binaria de les dades extretes de les corbes de Bézier i posterior importació.

Canvi de Format: Transformació de les dades extretes de les corbes de Bézier al format d'animació de Unity compatible amb la Unity Timeline.

Exportar: Transformació directe des de les dades extretes de les corbes de Bézier al format d'animació de Unity i posterior serialització.

Control de Càmera:

Moviment Lliure: desplaçar la càmera a través de l'entorn 3D i enregistrar el recorregut a partir del qual s'obtindran les corbes de Bézier.

Tracking amb moviment orbital lliure: molt similar al moviment lliure, la càmera seguirá l'objecte especificat i només es podrà moure la càmera lliurement en una órbita al voltant de l'objecte.

Càmeres estàtiques: Generació de càmeres estàtiques i reposicionament per trobar el millor angle de càmera.

Generació de camins: utilitzar Gizmos per definir punts i tangents en l'espai que generen un camí a partir de segments de corbes de Bézier.

Eines de support:

Pantalla de suport: Pantalla que visualitza la vista de la càmera seleccionada utilitzant una textura dinàmica.

Menu radial de càmeres: per canviar d'una càmera a una altre de forma senzilla i ràpida.

Extra:

Cutscene: exemple de cutscene creada amb l'eina desenvolupada.

Plugin: crear un package per poder publicar la versió definitiva a Unity.

1.5 Abast del projecte

El projecte concluirà amb un prototip d'eina que permeti gravar el recorregut de les càmeres en un escenari 3D i reproduir-lo en un videojoc 3D. Posteriorment aquest prototip es pot sotmetre a una fase de polit i millora de la interfície d'usuari amb l'objectiu de publicar-lo a la botiga de Unity.

Si es publica, aquesta eina es dirigirà a empreses petites i mitjanes que utilitzin Unity com a motor per crear videojocs o altres aplicacions. Els dissenyadors i artistes d'aquestes empreses la podran utilitzar per enregistrar cinemàtiques més immersives de forma ràpida i senzilla. Ajudarà a empreses a reduir tant els costos econòmics, com el temps de producció o la necessitat de tenir personal especialitzat. A la vegada obrirà una porta d'entrada per acostar persones d'altres sectors com el cinema a la indústria dels videojocs i relacionats.

2. Estat de l'art

2.1 Introducció a les Cutscenes

Les Cutscenes o cinemàtiques s'utilitzen com a eina narrativa per presentar: diàlegs entre personatges, l'entorn del jugador, obstacles o nous elements de gameplay entre altres.

El terme Cutscene va aparèixer per primer cop als anys 80 de la ma de Ron Gilbert i s'ha convertit en una de les eines més utilitzades dels dissenyadors per empènyer el jugador a un estat emocional més intens durant el temps de joc.

Amb els anys la capacitat tecnològica i els recursos a l'abast dels desenvolupadors no ha fet més que augmentar. Les eines de producció de cutscenes s'han desenvolupat adaptant-se a la tecnologia existent, definint diferents tipus de cutscenes segons la tecnologia o eines utilitzades.

Seguint aquesta evolució es poden distingir quatre tipus de cutscenes:

- **Live-Action:** film gravat amb actors reals que interpreten el paper dels personatges.
- **Pre-Rendered:** escenes generalment de millor qualitat gravades amb softwares especialitzats a les cuals s'afegeixen efectes de prost-processat i efectes especials (CGI).
- **Real Time:** escenes produïdes directament en el videojoc a través de Scripting.
- **Interactive Cutscenes:** escenes Pre-Rendered/Real Time que permeten una presa de decisions per part del jugador en moments clau per completar una acció.

En la següent imatge podem veure la diferència gràfica que existia fa uns anys entre una cinematica Renderitzada prèviament (esquerra) i una en temps real (dreta). Exemple del videojoc Final Fantasy XIII publicat a finals del 2009.

Figura 2.1: *Final Fantasy XIII. extreta de "Final Fantasy XIII - CGI vs RealTime Comparison"*, última consulta 05/2019.

2.2 Real Time Cutscenes

Les escenes en temps real són escenes generades per el motor del joc en temps d'execució. En aquestes escenes es limita el control que te el jugador sobre el joc i via scripting es reprodueix un guió predeterminat. Sovint s'utilitzen models amb més poligonatge i millors textures durant l'execució de la cutscene per obtenir un millor resultat gràfic. També s'inclouen animacions exclusives per adaptar les accions dels personatges al guió.

Cal destacar que la industria actual dels videojocs ha adoptat el Real Time o in-Game com a estàndard per produir les escenes cinematogràfiques dels videojocs. Aquesta tendència ve donada per els avantatges d'aquesta tècnica i la qualitat gràfica, cada cop amb més resolució, dels motors de videojocs, les consoles i els ordinadors actuals.

A pesar de no arribar al nivell de les escenes Pre-Rendered en quant a gràfics, disposen d'altres avantatges com la poca memoria requerida, la personalització dels personatges i la millor immersió dels jugadors al no haver-hi una diferència tan gran entre l'actual gameplay i la cutscene.

Un bon exemple d'aquesta aposta de la indústria per el Real Time és el videojoc "Uncharted 4 a thief's end", publicat el 2016, que només utilitza cinemàtiques en temps real.

Figura 2.2: *Uncharted 4. Extreta de l'article de d'Andrew Webster a the Verge on s'analitza Uncharted 4. ultima consulta 05/2019.*

2.3 Unity Engine

Unity Engine és un motor de videojocs multiplataforma que permet el desenvolupament de videojocs i altres continguts multimèdia. Actualment és el motor de videojocs més utilitzat en el món i engloba funcionalitats per a la creació de tot tipus de continguts.

Empreses de diversos sectors com són el dels videojocs, la industria motriu i del transport, la industria cinematografica o la industria de la construcció utilitzen aquest motor com a eina.

La versió estable més recent de Unity és la 2018.3[1] i esta disponible per Windows, OS X i Linux. Amb Unity es pot crear aplicacions per una gran varietat de dispositius com: ordinadors, tablets, mòbils, etc.

Enfocant el tema de les cutscenes, motors com Unity permeten als desenvolupadors abordar el tema des de varis punts de vista. Una de les opcions que té tant aquest motor com en la resta de motors és la programació via scripting dels recorreguts de les càmeres que s'utilitzaran per les cutscenes.

Una altre opció que tenen és la importació d'animacions de càmeres creats en altres softwares com Maya i la reproducció d'aquests clips a través de scripting o des de la mateixa timeline que proporciona Unity.

També hi ha l'opció d'utilitzar software de tercers. Unity disposa de la Asset Store, una botiga virtual on els desenvolupadors poden penjar o descarregar packages, ja siguin assets o plugins, de forma gratuita o pagant i utilitzar-los en els seus videojocs.

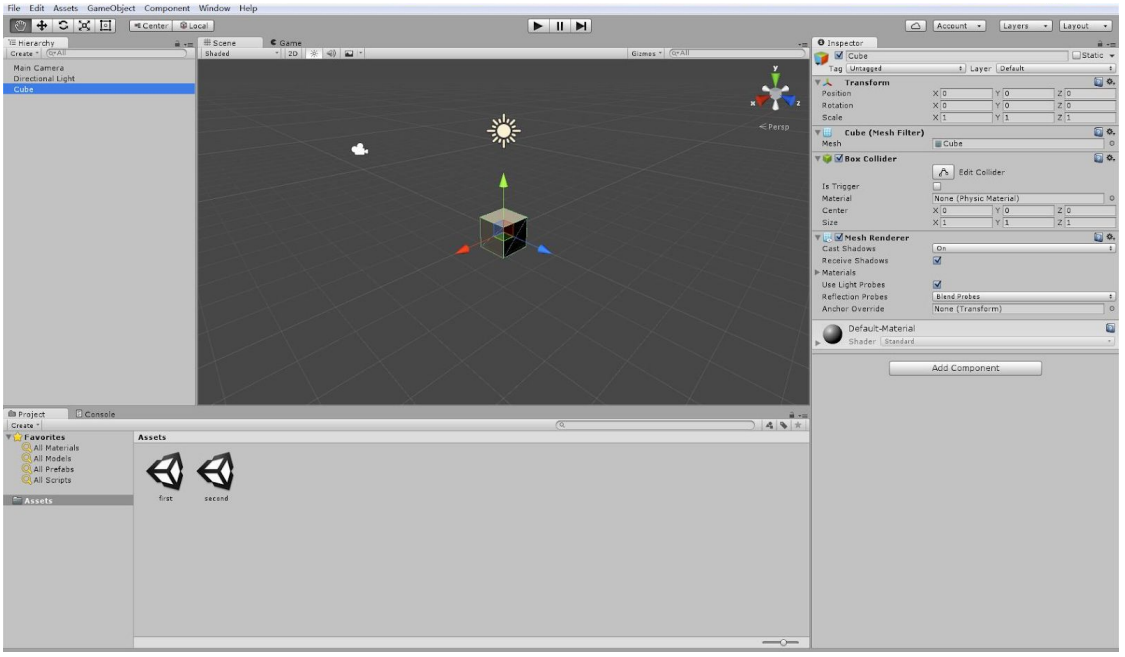

Figura 2.3: *Unity Editor, Imatge de creació pròpia.*

2.3.1 Timeline

La Timeline o línia de temps és un element molt utilitzat en tot tipus de software d'animació. Generalment consta de pistes o Tracks que es reprodueixen seguin una evolució del temps, ja sigui en framerate o en segons. Cada una d'aquestes pistes pot representar elements diferents, com son els personatges i les seves animacions, les animacions de les càmeres, clips d'àudio a reproduir, efectes de llum o sistemes de partícules.

Cada Track pot tenir un seguit de clips en el seu format corresponent: audio, animació, etc. Segons quin tipus de track sigui també permet transicions entre clips, utilitzant blending o similars.

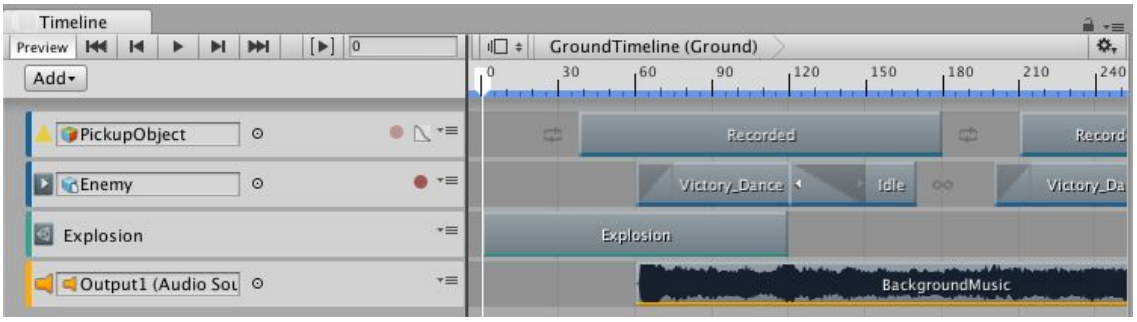

Figura 2.4: *Timeline. Extreta del manual de Unity: TImelineOverview última consulta 05/2019*

2.4 Realitat Virtual

La tecnologia de Realitat Virtual (VR) és una de les invencions més rellevants dels últims anys i ha agafat molt de protagonisme en diversos sectors com són el sanitari, el formatiu o el dels videojocs.

La Realitat Virtual és una tecnologia que es caracteritza per sumergir amb més o menys traça l'usuari en un entorn 3D. Aquesta immersió total en un entorn virtual permet als usuaris viure experiències increibles.

Degut a aquest grau d'immersió tan gran per part dels usuaris, molts sectors i empreses s'han interessat en produir productes per aquesta tecnologia, creant un catàleg d'aplicacions en expansió.

Unity, com altres motors, ha creat i adaptat eines per facilitar el desenvolupament de contingut per VR, apostant per la creació d'experiències immersives utilitzant el seu motor. El resultat d'aquesta aposta per la VR és que el 60% de tot el contingut de Realitat Virtual ha estat creat amb Unity.

Algunes de les eines a disposició dels desenvolupadors enfoquen les seves funcionalitats a millorar la interacció amb l'entorn 3D i el moviment o a crear cinemàtiques interactives amb audios 3D entre altres.

2.5 Estudi de Mercat

Per tenir una visió més àmplia del mercat objectiu s'analitzara Unity Engine i les eines disponibles dins del motor per a la creació de Cutscenes. Limitant l'abast a la funcionalitat de l'engine i plugins que es puguin fer servir com a eines de creació i edició de recorreguts de càmeres virtuals en entorns 3D.

Dels plugins disponibles a la tenda n'hi ha dos que solucionen o intenten solucionar el problema presentat.

2.5.1 Cinemachine

Cinemachine és l'opció més professional que hi ha actualment en el motor de videojocs Unity i està desenvolupat per Unity Technologies, propietaria del motor. Aquesta eina ha passat de ser un plugin gratuït de la tenda Unity a estar integrat directament en el motor.

És una eina al nivell de videojocs AAA que centra el seu funcionament en l'edició i manipulació de les càmeres presents a la escena 3D.

Figura 2.5: *Cinemachine. Extreta de la pàgina principal de Cinemachine última consulta 05/2019*

Els components i funcionalitats destacades de Cinemachine que es podrien tenir presents en el desenvolupament de l'eina són:

Component CM Virtual Camera:

En Unity només es permet tenir una càmera activa, les càmeres virtuals prenen la posició de la càmera activa per poder pintar des de la seva perspectiva en un moment determinat.

Aquestes CM Cameras Virtuals disposen de varies funcionalitats integrades de les quals només explicaré les que poden ser objecte d'investigació per aquest treball.

Composer:

Algoritme integrat a la component CM Virtual Camera que permet mantenir el focus de la càmera centrat en un objecte definit per els dissenyadors des de l'editor.

Explicat de forma senzilla, l'algoritme utilitza tres marges customitzables des de l'editor que separen la superficie de render de la càmera en tres zones. Depenent de la zona on es troba l'objecte en seguiment es modifica el comportament de la càmera, concretament la seva rotació.

Aquestes tres zones son:

- Death Zone: la càmera no es desplaça perquè considera que l'objectiu esta enfocat.
- Soft Margin: la càmera es desplaça en funció de quan de lluny està l'objecte de la Death Zone, segons els desenvolupadors de l'eina fa l'efecte d'una esponja presionada que torna a la seva forma inicial.
- Hard Margin: limit de l'esponja, fa la funció d'una paret que no deixa sortir l'objecte de la zona delimitada.

A efectes pràctics és una versió millorada del Billboard, un algorisme que alinea la transformació de la càmera per enfocar un punt en concret, transformació de l'objecte a seguir.

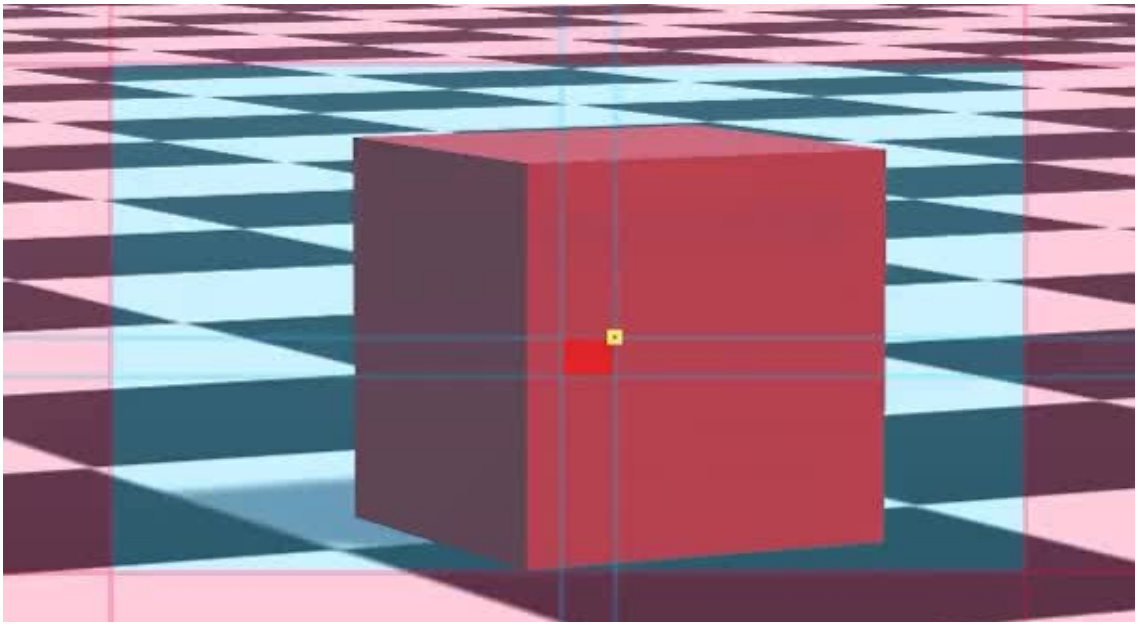

Figura 2.6: *Composer. Extreta de Cinemachine: Module Examples última consulta 05/2019*

Franja vermella: Hard Margin, Franja blava: Soft Margin, Quadre central: Zona Morta

Transposer:

Algoritme integrat a la component CM Virtual Camera, permet desplaçar la càmera en relació amb un objecte, mantenint una distància o offset determinat per el dissenyador entre l'objecte assignat i la càmera actual.

Cinemachine Path:

Es tracta d'un camí format per waypoints, una estructura que conté punts en l'espai i tangents. Amb aquest punts i tangents es poden formar corbes de Bézier mitjançant interpolació i utilitzar la unió d'aquestes corbes com un camí a seguir.

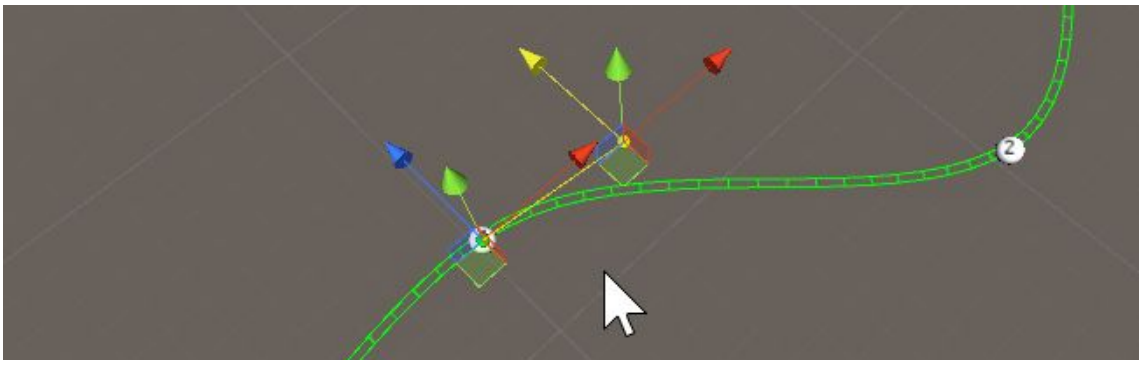

Figura 2.7: *Cinemachine Path: extreta del fòrum de Unity última consulta 05/2019*

Cinemachine Dolly Track:

El Cinemachine Dolly track utilitza Cinemachine Paths de guia per desplaçar la càmera en funció del temps especificat a la Timeline. També permet modificar la rotació Roll de la càmera al llarg del recorregut per enfocar una o altre direcció.

Cinemachine Tracked Dolly:

Molt semblant al Cinemachine Dolly Track amb la diferència que aquest segueix el camí en funció de la posició de l'objecte a seguir. A la pràctica buscarà sempre el punt del camí més proper a la transformació de l'objecte dins del Cinemachine Path assignat.

Figura 2.8: *Tracked Dolly: extreta del fòrum de Unity, última consulta 05/2019*

Blending:

Si es vol passar d'una camera virtual a una altre en un espai de temps específic, el blending permet fer servir una interpolació per fer una transició fluida entre les dues càmeres. La interpolació és pot fer utilitzant diverses corbes que donaran resultats diferents.

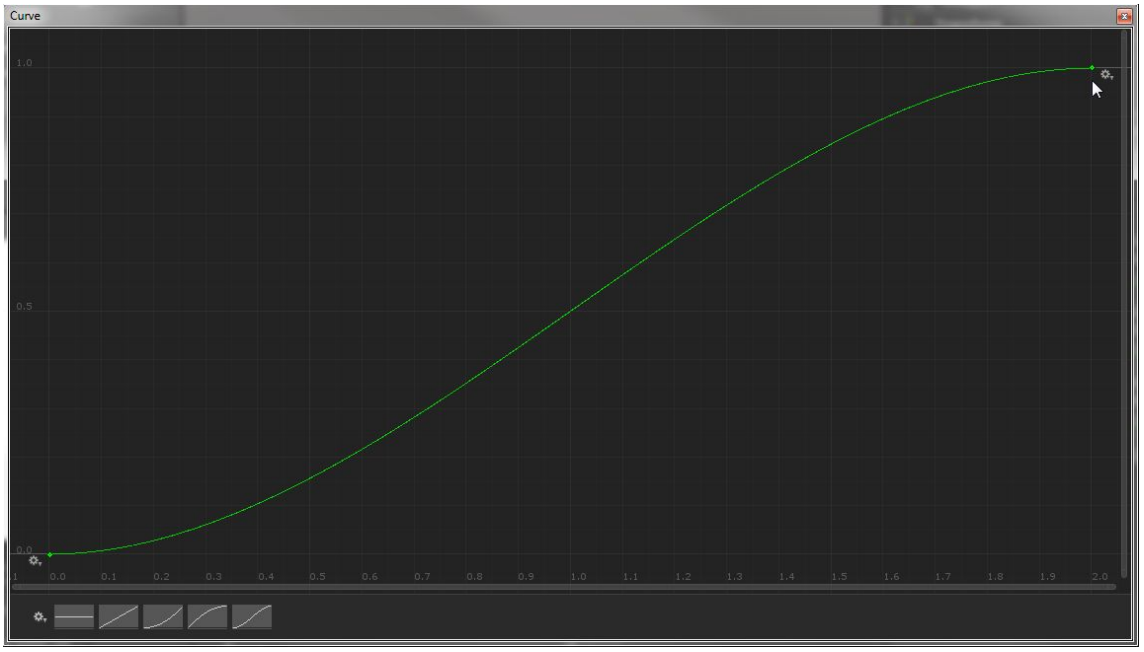

Figura 2.9: *Blending Curve: extreta de Cinemachine: Examples, última consulta 05/2019*

Algoritmes de Cinemachine a reprodüir:

En la implementació del projecte s'intentarà simular varies d'aquestes funcionalitats i esbrinar les matemàtiques que es porten a terme. Les funcionalitats que poden aportar un plus a l'eina a desenvolupar són les següents:

- Composer: mantenir l'objectiu en el focus de la càmera utilitzant soft margins.
- Transposer: mantenir una distancia determinada amb l'objecte a pesar del desplaçament.
- Cinemachine Path: aquest es definirà a partir del recorregut gravat amb les ulleres i els controladors de VR HTC Vive extrapolant punts i tangents a partir de segments formats per corbes de Bézier.
- Dolly Track: camí per on es desplaçarà la càmera en funció del temps de la timeline.
- Blending: transició fluida entre una càmera i una altre.

2.5.2 VR Cameraman

VR Cameraman és un plugin disponible a la Asset Store de Unity que explora el concepte d'utilitzar VR per gravar les animacions de càmera. Dins de les funcionalitats programades pels dos joves desenvolupadors destaca el fet de poder exportar i importar els recorreguts gravats com a AnimationClips, format d'animació estàndard de Unity, permeten utilitzar i editar les animacions directament des de la Timeline.

Figura 2.10: *VR Cameraman: extreta de l'Asset Store, última consulta 05/2019*

Aquesta eina també dona varies opcions per controlar la càmera, des de portar la càmera en mà com feria un camera no professional fins a posar varies càmeres en l'escena, fer interpolació entre dos punts o saltar d'una càmera a una altre de forma senzilla.

Com s'ha esmentat amb anterioritat, aquesta eina era una solució parcial al problema presentat ja que no esta operatiu amb les versions actuals de Unity, 2018 i posteriors, i tampoc s'ajusta a una execució desitjada. Això no vol dir que no tingui funcionalitats excel·lents que poden ser útils en l'eina a desenvolupar.

D'aquestes funcionalitats la que destaca i s'intentarà implementar es la següents:

Render de la Càmera:

Una de les funcionalitats presents en aquesta eina que s'inclourà en el prototip és la pantalla de suport que rendereja el que veu la càmera actual i utilitza el render com a textura en la part superior d'un cub, objecte representatiu de la càmera.

Aquesta funcionalitat és molt important si es vol utilitzar la VR, ja que en l'entorn de realitat virtual no pots accedir a la finestra d'editor de Unity o al camp de visió de la càmera que es vol editar.

Figura 2.11: *Record: extreta de l'Asset Store, última consulta 05/2019*

3. Gestió del projecte

3.1 Procediment i Eines per al seguiment del projecte

3.1.1 GANTT

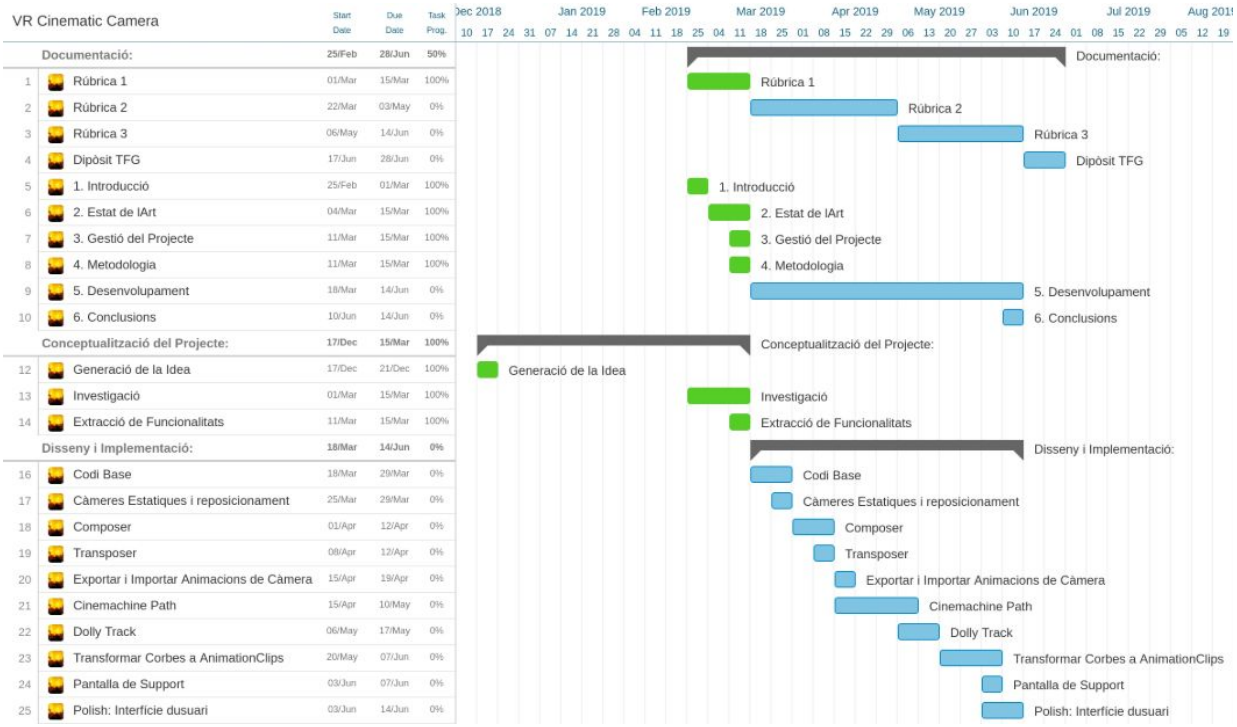

Figura 3.1: *Diagrama de Gantt. Imatge de pròpia creació.*

3.1.2 Trello

Trello és una aplicació web i d'escriptori que permet millorar el seguiment de tasques utilitzant taulells a on penjar notes relacionades amb les tasques i el seu estat.

Per tenir una millor organització de les tasques a desenvolupar, s'utilitza un taulell que classifica les tasques a realitzar en columnes segons el seu estat i prioritat. D'aquesta manera es podrà gestionar de forma senzilla totes les tasques, prendre notes, modificar l'estat de les tasques i tenir un registre de tots els problemes existents.

3.1.3 GitHub repositori en xarxes, Git eines de control de versions

Un altre de les eines habituals en el cas de desenvolupament de software és la utilització de Github. Github permet emmagatzemar el codi a la xarxa i mantenir un control de versions, ideal en projectes on els errors son dificils de percebre fins que és ja massa tard.

Per desenvolupar el projecte s'ha creat un repositori privat de Github dins de la compte particular del titular d'aquest treball.

3.2 Eines de validació

Validació de Funcionalitats: Es considerarà un desenvolupament exitos la creació d'una eina que tingui implementades les següents funcionalitats:

- Animacions de càmera utilitzant camins extrets a partir de corbes de Bézier creades o gravades amb l'eina.
- Interfície d'usuari que permet-hi una interacció satisfactòria amb l'eina i les seves funcionalitats.
- Algorisme de seguiment d'objectes similar al Composer de Cinemachine.
- Compatibilitat de les animacions creades amb la Timeline de Unity.

Validació d'usabilitat: Per validar la utilitat de l'eina a l'hora de solucionar el problema plantejat a l'inici del treball, es compararà l'animació de càmera creada amb VR Cinematic Camera i una animació de càmera d'un joc professional.

Al ser una validació molt subjectiva, el resultat sera positiu si:

- Cutscene d'exemple: creació d'una cutscene en temps real utilitzant l'eina desenvolupada.
- Comparar temps de creació d'una cutscene amb cinemachine i una amb l'eina desenvolupada.
- Moviment de càmera similar a una escena d'un joc triple A.
- Temps de creació relativament curt.

3.3. DAFO

Taula 3.3. Anàlisi DAFO.

	Positius	**Negatius**
Origen Intern	**Fortaleses** Experiència prèvia amb corbes de Bézier i animacions de UI. Comprensió del funcionament d'algoritmes com el Composite, Transpose i Tracks & Dollies.	**Debilitats** - Falta de capacitat tecnica a l'hora de desenvolupar algoritmes complexos com el composite o l'extracció dels waypoints a partir d'una corba de Bézier. - Dificultat per adaptar les animacions a la timeline.
Origen Extern	**Oportunitats** Gran augment de la utilització de la Realitat Virtual per part d'usuaris i desenvolupadors. Forat en el mercat, no hi ha eines operatives que utilitzin la VR com a eina per produir Cutscenes	**Amenaces** - No arribar a un nivell acceptable que converteix en útil l'eina si es compara amb la alternativa directa, Cinemachine. - Problemes legals per intentar simular funcionalitats ja existents a Cinemachine.

3.4. Riscos i pla de contingències

Durant el procés de producció de VR Cinematic Camera està exposat a diversos riscos que es poden classificar en funció de la seva tipologia.

Riscos per:

- Falta de capacitat tècnica:

Molts dels algorismes implementar requereixen d'una capacitat tècnica i coneixement matemàtic alt. Un problema afegit és el fet que el codi actual que es podria fer servir de referencia és privat i només se'n pot intuir la idea darrera les matemàtiques que executa.

Entre aquests algorismes hi ha el Composer, les corbes de Bézier i el render de textures dinàmiques.

- Falta de capacitat de disseny:

També hi ha la possibilitat de implementar un disseny que no sigui intuitiu o útil per als potencials usuaris de l'eina. Així com la implementació de funcionalitats que no aporten valor suficient com per utilitzar l'eina envers altres opcions.

Taula 3.4. Pla de Contingències.

Risc	Solució
Risc Tècnic: no poder implementar un algorisme similar al utilitzat pel Composer.	1. No implementar i utilitzar el component CM Virtual Camera de Cinemachine que té l'algorisme implementat. 2. Implementar la versió simplificada, Billboard.
Risc Tècnic: Extracció de Corbes de Bézier	1. Segmentar el camí en punts cada "x" temps i fer interpolació lineal entre ells.
Risc Tècnic: Algorisme per crear Textures dinàmiques	1. No implementar la funcionalitat
Risc Tècnic: No poder implementar compatibilitat entre les animacions de càmera a la Timeline de Unity.	1. Sistema que gestioni la sincronització de les animacions amb la Timeline a partir d'events.
Risc Disseny: Interfície d'usuari poc intuitiva o útil.	1. Solicitar el suport d'un dissenyador per re-dissenyar la interfície d'usuari.
Risc Disseny: Implementació de funcionalitats poc valorades per els usuaris.	1. Demanar suggerències de com millorar funcionalitats ja implementades o per afegir-ne de noves.

3.5. Anàlisi inicial de costos

En la **taula 3.5.1** Costos es pot veure els costos derivats d'aquests projecte. En l'apartat de recursos humans (RR.HH), es fa una distinció dels rols necessaris per desenvolupar el projecte i les hores dedicades a cada una d'aquestes funcions. Els costos inclouen les despeses a que es pot incórrer durant els 4 mesos de desenvolupament i ascendeix a un total de aproximadament 5.500 €.

		Any 2019			
Tipus de Cost	Desglossat	Març	Abril	Maig	Juny
RR.HH					
	Disseny i Documentació	270,00 €	- €	- €	- €
	Programador	360,00 €	630,00 €	450,00 €	360,00 €
	Programador UX	- €	- €	180,00 €	270,00 €
	Dissenyador UX	- €	- €	150,00 €	50,00 €
Despeses d'oficina i Altres					
	Lloguer de despatx	275,00 €	275,00 €	275,00 €	275,00 €
	Cafe i Aperitius	120,00 €	120,00 €	120,00 €	120,00 €
	Material d'Oficina	5,00 €	- €	- €	- €
Despeses Software					
	Github	7,00 €	7,00 €	7,00 €	7,00 €
Equipament i Amortització					
	PC, displays, etc	34,17 €	34,17 €	34,17 €	34,17 €
	HTC Vive	17,22 €	17,22 €	17,22 €	17,22 €
Marketing					
	Webs especialitzades	- €	- €	- €	1.000,00 €
Total Costos					
	Total	1.088,39 €	1.083,39 €	1.233,39 €	2.133,39 €
	Acumulatiu	1.088,39 €	2.171,78 €	3.405,17 €	5.538,56 €

En la **taula 3.5.2** Desviació de Costos es mostra la possible desviació dels costos del projecte en funció de tres escenaris: escenari ideal, escenari estimat i escenari dolent. A la vegada es separa la desviació en dos tipus, variació de la quantitat d'hores, i augment d'altres costos com el lloguer de l'oficina, material i aperitius. Es considera estable el cost de software, equip, amortització i Marketing.

	Escenaris possibles		
	Ideal	Estimat	Dolent
Desviació Hores	100%	125%	150%
Desviació Altres	100%	110%	120%
Cost amb Desviació	5.538,56 €	6.377,06 €	7.215,56 €

3.6 Modificacions en la planificació:

Durant el procés de desenvolupament del projecte entre els dies 15/03/2019 i 3/05/2019, s'han produït canvis en la planificació per tal de millorar el procés i obtenir un millor resultat final.

Aquests canvis són deguts a un canvi en l'estrategia de desenvolupament del projecte i només han afectat a dues tasques de la següent manera:

1. Cinemachine Path: Tasca 21 del diagrama de Gantt.

En l'etapa de disseny de la funcionalitat, es va observar que la tasca es podia dividir en dues sub tasques definides de forma molt clara: l'algorisme que defineix i crea el camí del Cinemachine i la interacció de l'usuari amb el camí per editarlo.

Com a tal la tasca s'ha dividit en dues, Creació de Cinemachine path (Bézier Path) i Edició del recorregut (Edit Bezier Path). Aquest canvi ha estat seguit per una modificació en els períodes de temps on es desenvoluparan les dues tasques. El canvi principal ha estat el de posposar l'edició del recorregut fins a les últimes etapes de desenvolupament ja que és quan es du a terme el disseny i desenvolupament de la interfície d'usuari.

El motiu de programar l'edició del camí en paral·lel amb la interfície d'usuari és que hi ha certa dependencia entre les dues.

Per tal de no afectar la data de finalització ni el pressupost plantejats es durà a terme un segon canvi en la planificació.

2. Transformar corbes de Bézier a Animation Clips: tasca 23 del diagrama de Gantt.

La tasca consisteix en convertir els camins creats en la tasca 21: Cinemachine Path a un format d'Animation Clips que després es guarda i es pot utilitzar com a animación de càmera.

S'ha escollit fer el canvi entre l'edició i aquesta tasca per la dependencia directa que té l'última amb la de creació de camins. Així s'ha considerat més productiu agrupar les tasques per facilitar el bon funcionament dels dos algoritmes i a la vegada agrupar les dues tasques que basen el seu funcionament en la interfície d'usuari i la interacció amb els controladors de realitat virtual.

3.6.1 Afectacions:

Com ja s'ha esmentat, aquesta modificació en la planificació no tindrà afectacions en el procés de desenvolupament més enllà del canvi en l'ordre de producció de les tasques. Les hores a dedicar a les dues tasques que s'intercanvien són aproximadament les mateixes per tant tampoc hi haurà retards en el desenvolupament.

Des del punt de vista dels pressupostos, les hores a dedicar a les dues tasques ja estaven contemplades per tant no afectarà al cost total de desenvolupament.

3.6.2 Gantt

Així quedaría el diagrama de Gantt a dia 3/05/2019 considerant el progrés realitzat i els canvis en la programació de les tasques.

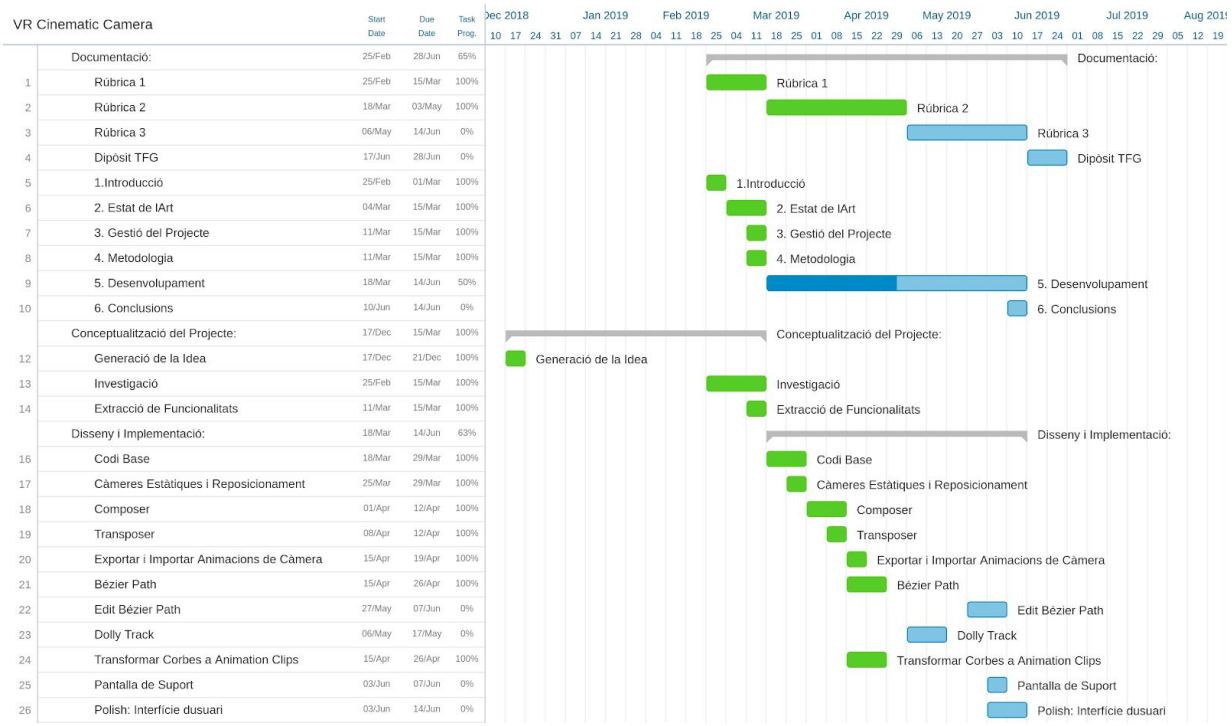

Figura 3.2: *Diagrama de Gantt Rúbrica 2. Imatge de creació pròpia.*

3.7 Avaluació de la Planificació

Fins les últimes setmanes de desenvolupament, el projecte s'ha desenvolupat amb normalitat. Tenint en compte les modificacions especificades en l'apartat 3.6, s'ha aconseguit dur a terme gran part del treball segons el previst.

No obstant, en el període de temps comprès entre els dies 27/05 al 12/06 la quantitat d'hores que s'han pogut dedicar al projecte han set moltes menys de les que estaven previstes. Per solucionar aquest contratemps s'ha mogut la tasca de creació de la pantalla de support, programada per aquells dies, a l'última setmana de desenvolupament.

La modificació de la planificació referent a la tasca de pantalla de support, no suposa cap desviació en el costos del projecte ja que a la pràctica a suposat moure les hores de treball contemplades en l'anàlisi de costos a l'última setmana de desenvolupament.

3.7.1 Gantt Final del Projecte

Així quedaría el diagrama de Gantt a dia 25/06/2019, data de finalització del projecte. S'ha actualitzant el progrés realitzat i afegit els canvis en la programació de les tasques esmentat anteriorment.

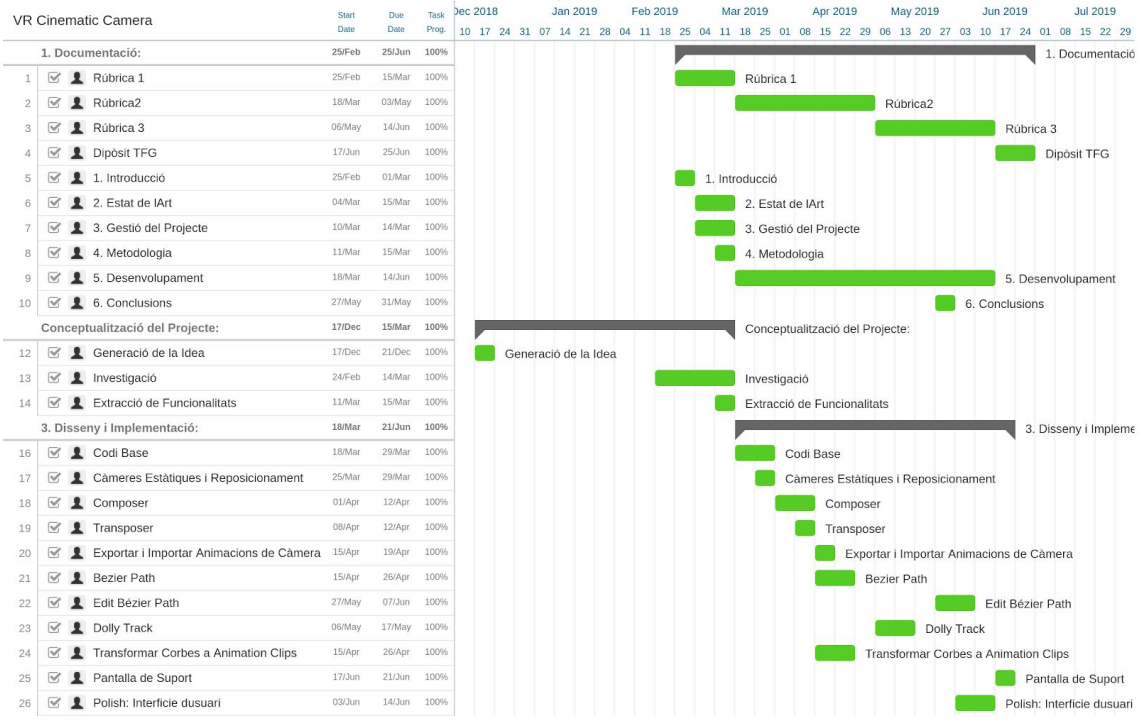

Figura 3.3: *Diagrama de Gantt Dipòsit TFG. Imatge de creació pròpia.*

4. Metodologia

En el projecte s'utilitzarà una de les metodologies més habituals del desenvolupament de software i derivada de les metodologies Agile. Aquesta metodologia es centra en el desenvolupament per funcionalitats, i el procés es coneix com a Feature-Driven Development (FDD).

El concepte és el de, a partir d'una concepte general de l'aplicació a desenvolupar, es focalitza els esforços en les funcionalitats a implementar, un cop implementades es testegen i es realitza una iteració fins aconseguir el resultat esperat.

4.1 Feature-Driven Development

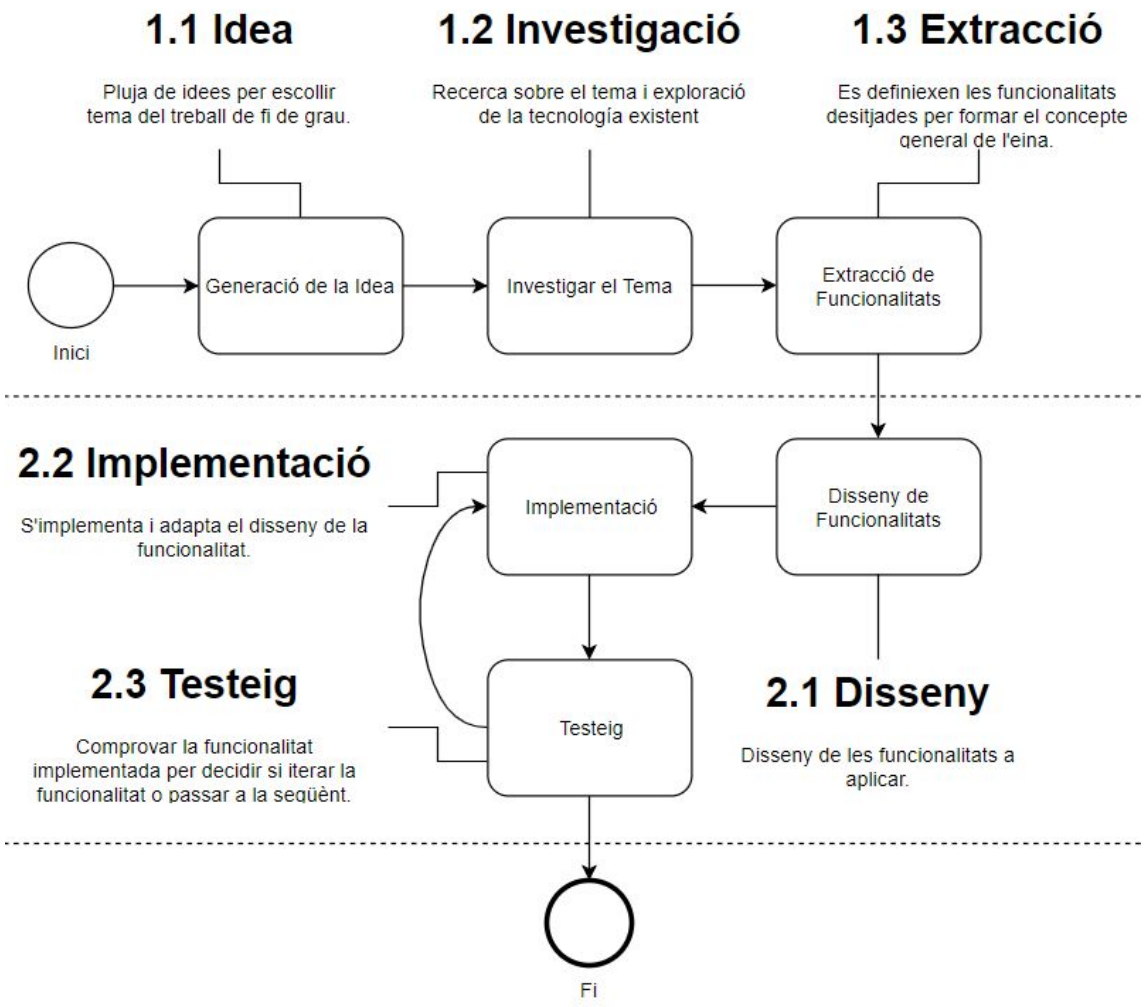

Figura 4.1: *Metodologia FDD. Imatge de creació pròpia.*

***Nota:** A dia 03/05/2019 es segueix utilitzant la metodologia FDD.*

4.1.1 Fases del Desenvolupament

El procés consta de les següents fases i subfases de desenvolupament:

1. Conceptualització del Projecte:

Donar forma al projecte, consta de les següents subfases

- Generació de la Idea:

Concepció de la idea basica i perfilació de les funcionalitats a implementar. Una exploració a un nivell molt superficial per tenir un mínim coneixement del tema i estimar la viabilitat del projecte.

- Investigació:

Fase del procés on s'explora amb profunditat el tema, s'investiguen les posibles aproximaciones al tema i com abordar des d'una nova perspectiva la realització de Cutscenes. En el marc d'aquesta investigació s'ha prestat molta atenció a les funcionalitats tècniques de Cinemachine i al concepte de VR Cameraman.

- Extracció de funcionalitats:

A partir de la investigació anterior, es defineixen quines son les funcionalitats que poden tenir cabuda en l'eina a desenvolupar. S'ha optat per agafar funcionalitats d'eines ja existents en l'entorn de Unity explorades en l'investigació previa i crear unió dels conceptes. Les eines ja esmentades són Cinemachine i VR Cameraman.

2. Disseny i implementació de Funcionalitats:

procés repetitiu per implementar les funcionalitats extretes en la fase previa.

- Disseny de funcionalitat:

En aquest subfase es desenvolupa la idea ja conceptualizada per adaptarla a la implementació esperada. Com adaptar una funcionalitat a l'eina en qüestió, definir característiques específiques i el plantejament personal de la mateixa.

- Implementació:

S'aplica el disseny anterior, es programa la funcionalitat i l'interfície d'usuari pertinent i s'ajusta al conjunt de l'eina per permetre una correcte interacció amb l'usuari.

- Testeig:

Subfase on es posa a prova la funcionalitat implementada per comprovar la seva usabilitat, en cas de no complir amb el resultat esperat es torna a la subfase anterior. Si el resultat és satisfactori es passa a la fase de Disseny i Implementació de la següent funcionalitat a implementar. Un cop implementades totes les funcionalitats es considera que l'eina esta completa i es surt del cicle per presentar l'eina ja finalitzada.

5. Desenvolupament del Projecte

Al ser Unity un motor que treballa amb gameobjects i components és vital dissenyar l'eina tenint en compte aquest aspecte. Per fer compatible l'eina amb aquest sistema, l'estructura de l'eina ha de separar les diferents funcionalitats en diversos objectes i scripts. Aquesta fragmentació de codi permet simplificar la comprensió de les funcionalitats, agrupar el contingut segons convingui i alleugerir la complexitat de integració de codi.

Per tenir clar en tot moment quina estructura mantenir, és important crear un esquema UML que defineix l'estructura de l'eina segons els criteris descrits. L'esquema dissenyat ajuda a mantenir una coherencia en el procés de disseny i desenvolupament de funcionalitats.

Esquema UML de referència:

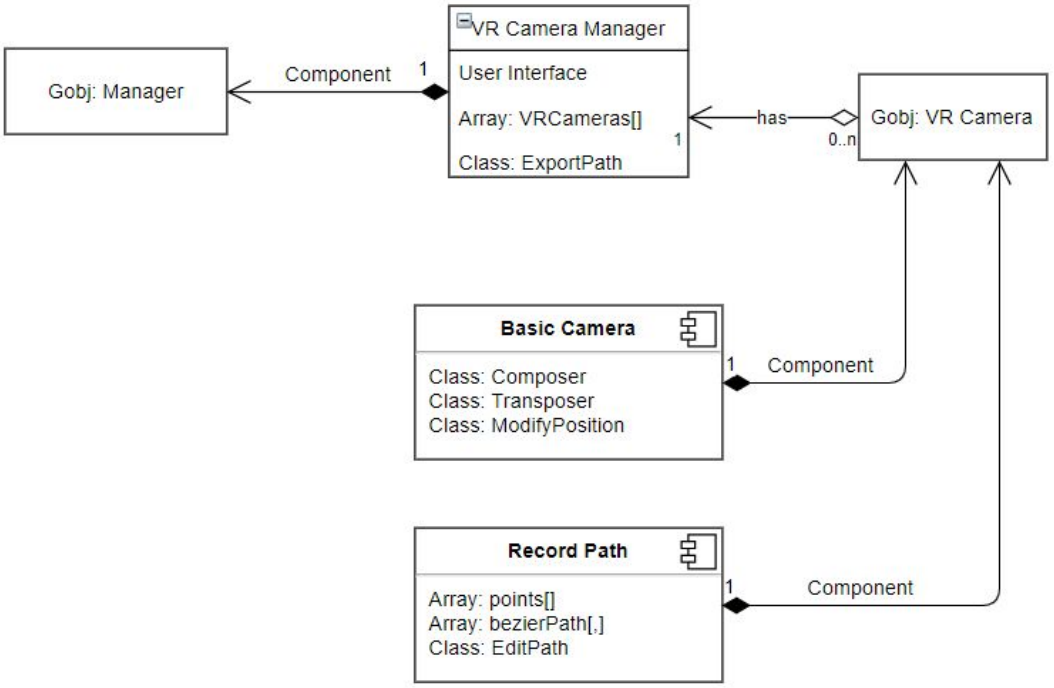

Figura 5.1: *Disseny UML de l'Eina. Imatge de Creació pròpia.*

Aquest esquema és el punt de partida per dissenyar un "Codi Base" que s'anirà ampliant a mesura que s'implementen les funcionalitats i representa l'estructura simplificada que hi haurà en tot moment a l'escena.

Aquesta estructura està formada per un objecte Manager que conté una component VR Camera Manager. El VR Camera Manager es la classe que gestiona l'eina i conté: una llista de VR Cameras, les funcionalitats troncals de l'eina que no necessiten estar en cada càmera de l'escena i gran part de la interfície d'usuari.

Entre els components de les VR Camera s'hi troben funcionalitats més específiques com són les de posicionament de càmeres, algorismes de composer i transposer o els camins a recórrer i els editors de curves.

5.1 Codi Base:

El disseny del codi base a implementar a la fase inicial sereix, de forma temporal, per poder testejar i utilitzar les funcionalitats desenvolupades, així com per definir una estructura que será la columna vertebral de l'eina.

Com s'ha esmentat aquest codi es modifica durant tot el projecte per tal d'incloure-hi les funcionalitats implementades. La base des de la que es parteix està formada per dos scripts principals.

Els scripts que formen part d'aquest codi base són:

5.1.1 VR Camera Manager

Al haver-hi una descentralització de funcionalitats és necessària la creació d'un gestor o manager que s'encarrega d'organitzar i gestionar les càmeres.

El VR Camera Manager és l'script que pren el paper del gestor, peça central de l'eina, permetent la utilització de les diferents funcionalitats per tal de gravar animacions de càmera.

En tota escena de Unity només hi pot haver un component Manager, situat en qualsevol gameobject.

Algunes de les funcionalitats que assumeix el gestor són les següents:

- Crear i eliminar càmeres predeterminades: Camera Estatica, Camera Bàsica o Càmeres amb Cinemachine Paths.
- Mantenir un llistat de totes les càmeres pròpies de l'eina presents a l'escena.
- Crear i exportar les Animacions de càmera
- Interfície d'usuari principal.
- Gestionar el comportament de les càmeres.

5.1.2 Basic Camera

La clase BasicCamera defineix el comportament base de les càmeres de realitat virtual, VR Cameras. En aquesta classe s'ajunten les diferents funcionalitats en quant al comportament propi de les càmeres.

Els comportaments principals incorporats són el composer, el transposer i la mecànica de reposicionament de la càmera. Els desenvolupadors poden interactuar amb aquesta classe des de l'inspector de Unity o desde la interfície d'usuari del VR Camera Manager per modificar les variables que afecten al comportament de la càmera.

L'inspector mostra un seguit de variables públiques a definir per els editors. En un primer fragment es decideix si incorporar un GameObject a seguir i si utilitzar composer i/o transposer. Els paràmetres propis de les classes Composer i transposer estan separats en pestanyes colapsades sota de les variables anteriors.

5.2 Basic Camera: Funcionalitats

En aquest apartat s'exposen les classes i algorismes presents dins la classe Basic Camera, així com el seu funcionament i la manera en què s'han implementat o adaptat.

5.2.1 Reposicionar càmeres

La creació i posicionament de les càmeres en una escena es pot fer de dues maneres. La primera és de de l'editor, creant una instancia de l'objecte VR Camera i situant-lo en la posició desitjada modificant de de l'inspector la component transformació. La component Transformació ésta present en tots els gameobjects i defineix la seva posició, rotació i escala dins l'escena.

La segona opció és la de posicionar l'objecte mentre s'executa l'aplicació. En aquesta segona opció és on entra en joc la funcionalitat de posicionament de l'script.

El principi és molt basic, en el moment de crear la càmera, aquesta agafa la posició i rotació del controlador dret (HTC Vive Controller) i les pren com a pròpies. Si l'usuari vol pot mantenir apretat un botó, de moment no especificat quin, per arrossegar l'objecte a la nova posició del controlador fins obtenir la posició desitjada o una aproximació.

A partir de la posició aproximada obtinguda es pot afinar de forma més precisa la posició i rotació de la càmera. En aquest punt s'utilitzaran dues maneres més precises, una per afinar la posició i l'altre per la rotació, amb l'objectiu final de situar la càmera perfectament.

1. Modificar la Posició:

Per moure de forma més precisa un objecte en realitat virtual utilitzarem un punt de pivot, la distancia del controlador al punt de pivot i una variable atenuant que servira de velocitat de desplaçament.

El funcionament d'aquest mètode consisteix en guardar la posició del controlador dret en el moment en que s'activa el posicionament de precisió. Aquesta posició de l'espai que es guarda serveix de pivot per a calcular el moviment.

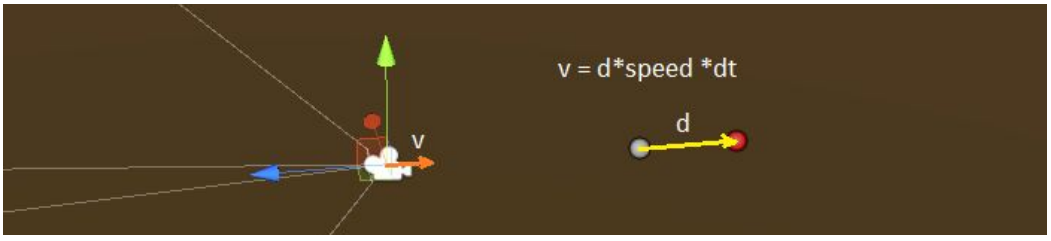

Figura 5.2: *Posicionar Càmeres amb Precisió. Imatge de Creació pròpia.*

Quan més s'allunyi la posició actual del controlador de la del pivot, més gran serà el desplaçament (v) a aplicar a la càmera. Cada frame, mentre la funcionalitat estigui activa, a la posició actual del controlador se li resta la del pivot, resultant en un vector direcció (d) amb una força determinada. Abans de moure la càmera en la direcció calculada es multiplica el vector per la velocitat i la derivada del temps per moure amb suavitat l'objecte.

Amb aquest mètode es pretén evitar un posicionament erroni de la càmera degut a possibles tremolors, moviments involuntaris o falta de precisió de la mà que subjecta el controlador. A més de minimitzar els efectes anteriors, al tenir en compte la derivada del temps, assegurem certa independència del framerate de l'ordinador cosa que resulta en un moviment més fluït.

2. Modificar la Rotació:

A més de la posició, també es pot ajustar la rotació de la càmera, per augmentar la precisió a l'hora de rotar la càmera s'utilitza un mètode similar a l'anterior.

Cal recordar que tota component transformació de Unity té el seu propi eix 'x', 'y' i 'z' que pertanyen a la transformació local de l'objecte respecte el món. Per girar amb més precisió l'objecte càmera, utilitzarem aquest eixos locals 'x', 'y' i 'z', o com els anomena Unity 'right', 'up' i 'forward', a més de la posició de la càmera, la posició inicial del controlador i una velocitat.

Al igual que en el moviment anterior s'utilitza el vector distancia entre la posició inicial del pivot i l'actual per calcular la rotació a aplicar sobre els eixos 'y' i 'x' de la càmera. Amb la distància que separa els dos punts s'obté el vector velocitat, l'angle de rotació serà proporcional al vector velocitat. Un cop tenim el vector velocitat, s'agafa el valor de 'y' com a velocitat de rotació sobre l'eix 'x' de la càmera, es multiplica per la derivada del temps i una variable speed i ja es pot aplicar la primera rotació.

Per fer la rotació en l'eix 'y' és necessari saber la magnitud del vector distancia(x,z) i el sentit de la rotació. També és necessari calcular el vector que va des del pivot a la càmera i fer el cross product amb el vector distància per saber si el sentit de la rotació ha de ser positiu o no.

Quan es té la magnitud de la velocitat('x','z') i el sentit de la rotació, es multiplica la magnitud per la derivada del temps i per la variable speed. Al aplicar les dues rotacions el resultat és una rotació precisa i fàcil d'utilitzar.

5.2.2 Composer

El composer és una de les funcionalitats principals del projecte i consisteix en fer el seguiment d'un gameobject rotant la càmera per tal de mantenir-lo sempre dins del seu focus.

El composer o compositor és un algorisme que intenta simular el moviment d'un càmera que sense moure's d'on està, només girant sobre si mateix mantingui enfocat un objecte amb la seva càmera.

Aquest algorisme també simula el lleuger lapse de temps que es produeix des del moment en que l'objecte es mou fins el moment en que el camera comença a rotar per mantenir l'objectiu dins el camp de visió de la càmera. El delay intensionat permet donar un punt extra de realisme a l'animació al ser un fenomen natural present en totes les produccions audiovisuals i televisives.

Precisament per aquest punt extra de realisme s'ha optat per implementar aquest sistema que, a pesar de ser més complex, resulta en unes cutscenes de més qualitat.

Altres opcions de més a menys qualitat són: enfocar la càmera a la posició de l'espai on estava l'objecte en el frame anterior o directament enfocar la càmera a la posició actual de l'objecte. No obstant les dues opcions tenen els seus inconvenients, en el primer cas, és possible que l'objecte surti del camp de visió de la càmera, i en el segon, el moviment resulta antinatural des del punt de vista dels jugadors.

1. Introducció a les Càmeres de Unity:

Per entendre exactament que es el que pasa en el composer primer hem d'entendre el funcionament basic d'una càmera a unity. Les càmeres de unity són gameobjects situats en un espai 3d amb una orientació determinada.

A una distancia "x" en la direcció "forward" de la transformació hi ha el "Near Clip Plane", un pla que representa el viewport i a on es projecten els objectes per extreure la imatge que es veu des de la càmera. Més enllà del "near plane" a una distancia de "x" vegades forward hi ha el "Far Clip Plane" que delimita l'espai que es projectarà sobre el Viewport. Aquests dos plans mantenen unes proporcions determinades pel FOV i si es creen plans entre les arestes corresponents es forma el frustum de la càmera.

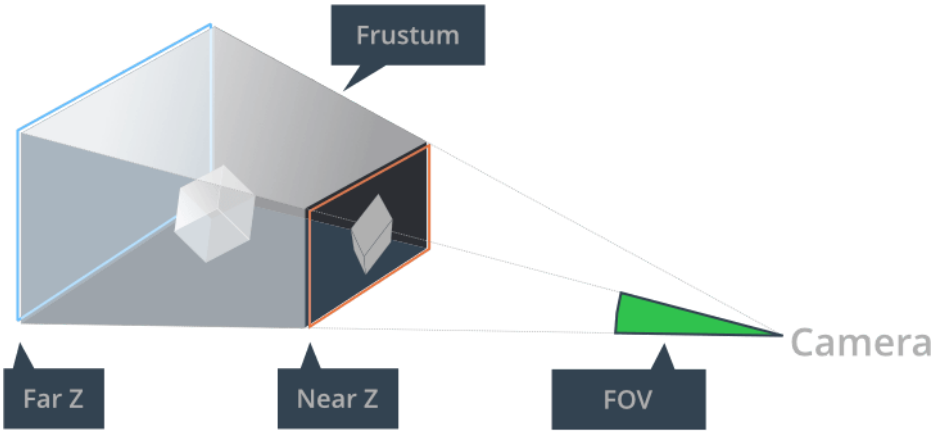

Figura 5.3: *Funcionament d'una Càmera: extreta de defold Manuals, última consulta 05/2019*

Tot el que estigui en aquest espai es considera dins del focus de la càmera i si no esta obstruït és visible en pantalla al utilitza la càmera com a origen del renderitzat. Per tant l'objectiu és que la posició de l'objecte a seguir estigui dins del Frustum i en conseqüència del viewport.

2. Classe Composer:

Com ja s'ha comentat a l'estat de l'art, l'objectiu és crear una aproximació al composer de Cinemachine. Seguint aquest com a exemple, es definiran dos rectangles en pantalla que defineixen el comportament de la càmera. El primer sera el HardRect, un rectangle que marca un marge rigid que l'objecte no pot creuar. El segon es el Softrect, un rectangle més petit i interior al HardRect que servira d'esponja per, en el moment en que l'objecte traspassi els límits, rotar la càmera en la direcció de l'objecte.

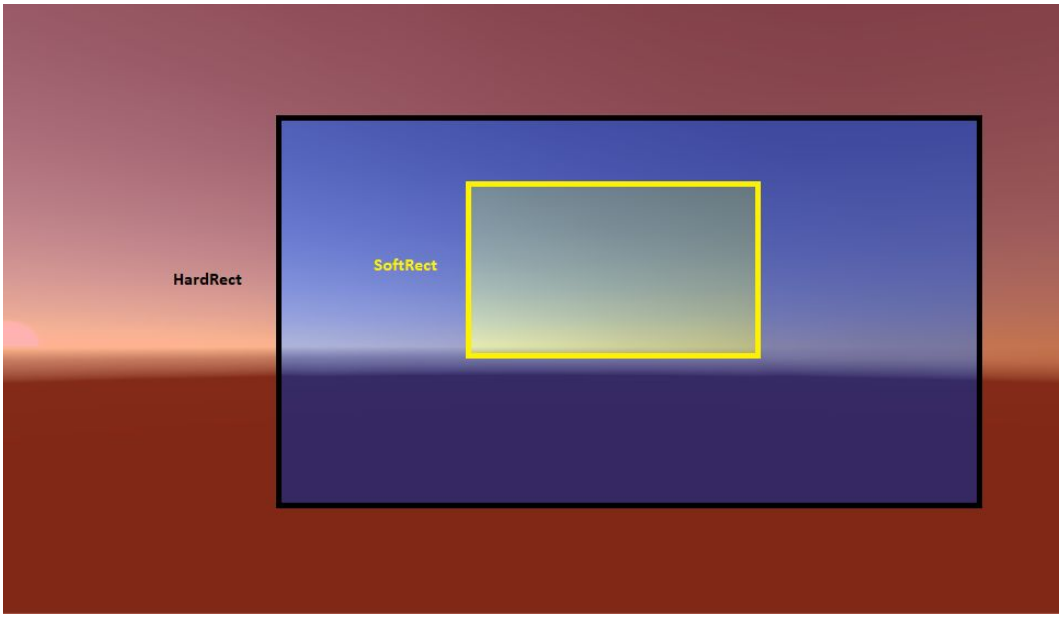

Figura 5.4: *Composer: Hard i Soft Rects. Imatge de Creació pròpia.*

Aquest dos rectangles marcaran el comportament de la càmera en tot moment. Separant la reacció de la càmera en tres comportament molt diferenciats.

Començant ja amb l'algorisme, es parteix d'un punt en l'espai 3d (point3D), posició de l'objecte a a seguir, i un gameobject amb la component camera, càmera activa que seguirà l'objecte.

Cal destacar que en tot moment s'està parlat de coordenades de pantalla es a dir pixels per calcular la rotació necessària de la càmera segons convingui.

El primer pas consisteix en projectar el punt de l'espai a coordenades de pantalla. Per fer aquesta projecció, existeix la funció de càmera "Camera.WorldToScreenPoint(point3D)" que donat un punt a l'espai torna la projecció sobre el pla de la càmera en coordenades de pixels.

Cal recordar que les coordenades de pantalla comencen en l'extrem inferior esquerra de la pantalla sent el (0,0) pixels i acaben a l'extrem superior de la pantalla com a (amplada, altura) pixels. Amb aquest punt ja es pot saber si l'objecte en concret esta fora o no de la pantalla, pero no es farà aquesta comprovació tenint en compte el rectangle Viewport(0,0, amplada, altura) sino amb els rectangles que s'han definit.

Els passos a seguir de l'algorisme son:

1. Projectar el punt a coordenades de pantalla.
2. Si el HardRect no conté el punt s'analitza el primer cas.
3. Si el HardRect el conté pero el SoftRect no el conte s'analitza el segon cas.
4. Si no és cap dels dos es descarta rotar la càmera, tercer cas.

Segons la franja en que estigui el punt projectat es presenten els tres casos esmentats anteriorment.

```
targetPos = camera.WorldToScreenPoint(target.transform.position);

if (!hardRect.Contains((Vector2)targetPos))
{
    HardZonePoint();
}
else if (!softRect.Contains((Vector2)targetPos))
{
    SoftZonePoint();
}
```

Figura 5.5: *Composer: Update Transform. Imatge de Creació pròpia.*

1. **Punt exterior al HardRect.**

El primer cas es el d'un punt que està fora del HardRect, és a dir el HardRect no conté el punt. El comportament establert per aquesta franja indica que l'objecte en cap cas pot traspassar el limits vermells, per tant l'objectiu de l'algorisme sera decidir quina rotació aplicar a la càmera perque el punt torni a estar dins dels marges acceptables.

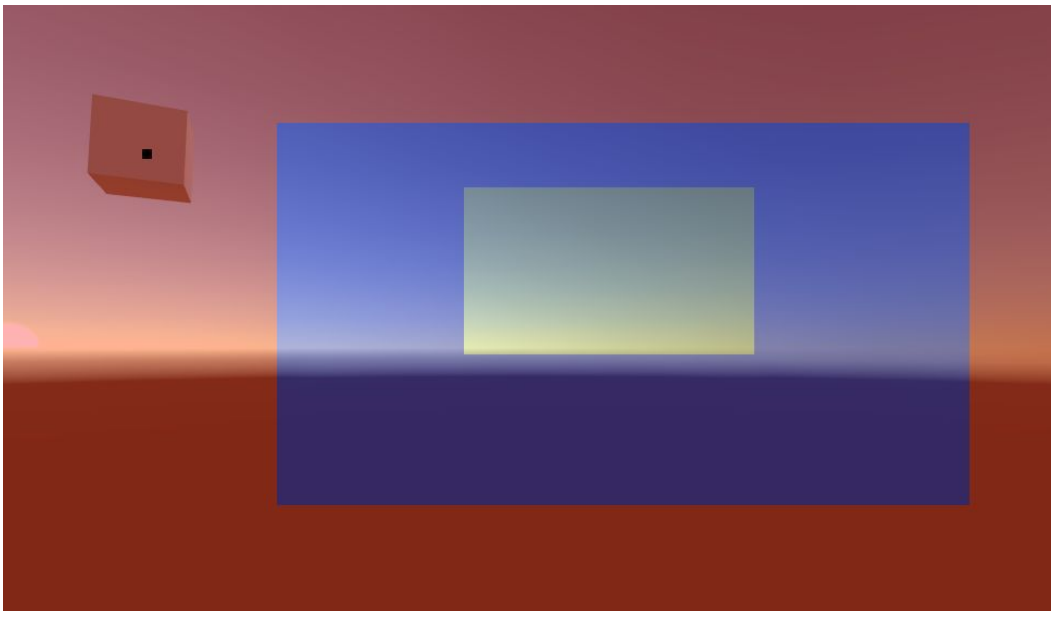

Figura 5.6: *Composer: Punt exterior al HardRect. Imatge de Creació pròpia.*

Donat el cas de que el punt està fora dels marges del HardRect, es calcula el vector que uneix el punt2D amb el centre del rectangle HardRect i obtenim el vector distancia. Amb el vector distància i fent uns càlculs simples podem saber en quin punt intersecciona la línia que uneix els dos punts amb algun dels límits del HardRect. si restem el vector intersecció de la línia que uneix els dos punts s'obté el desplaçament "extra" que fa que el punt estigui fora del rect.

Per últim, tenint en compte que s'utilitza la funció de càmera LookAt(Point) que centra la càmera en un punt. Cal saber quin és aquest punt, primer es calcula el centre de la pantalla en pixels més el vector desplaçament extra i es transforma a coordenades de món. Les coordenades de món resultants són el punt on s'ha de centrar la càmera perquè l'objecte es situï en el punt on intersecciona el vector distància amb el HardRect.

2. **Exterior al SoftRect, interior al HardRect.**

El segon cas és molt força similar al primer i passa quan el punt projectat està entre el HardRect i el SoftRect, és a dir en la franja blava. El comportament establert per aquesta franja indica hi ha d'haver una atracció del SoftRect que intenta mantenir el punt en el seu interior (zona transparent). L'objectiu de l'algorisme en aquest cas sera decidir amb quina força es sent atreta la càmera cap a l'objecte per aplicar una rotació proporcional a aquesta força. Si l'objecte esta quiet en aquesta franja la camera anira rotant amb una força proporcional a la distancia que falta perquè l'objecte acabi dins dels marges adients.

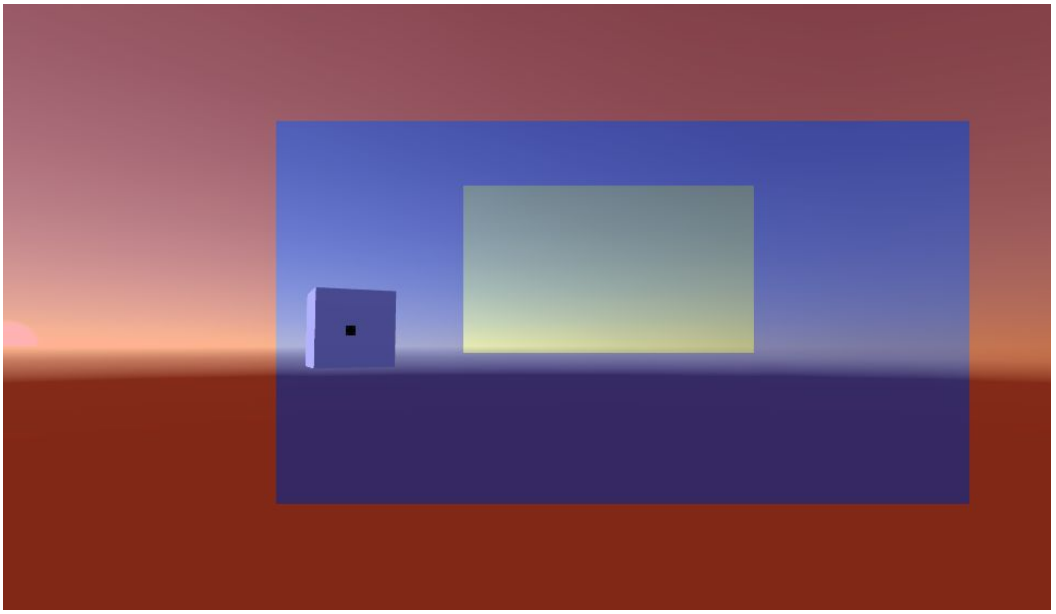

Figura 5.7: *Composer: Punt Exterior al SoftRect. Imatge de Creació pròpia.*

Com ja s'ha comentat aquest cas és molt similar al primer en quant a com calcular el desplaçament extra del punt2D fora de la SoftZone. Un cop es té aquest desplaçament extra s'utilitza una variable Kforce que indica quina és la amortiguació de la força. Quan més petit és el valor de Kforce més "rígida" és la SoftZone. Sent 0 un valor equivalent al de la HardZone que no deixa passar els objectes.

3. **Interior al SoftRect.**

Per últim es presenta el cas més simple, el punt està dins dels marges desitjats del SoftRect. En aquest cas la càmera considera que l'objecte esta en seu focus i descarta realitzar cap rotació extra.

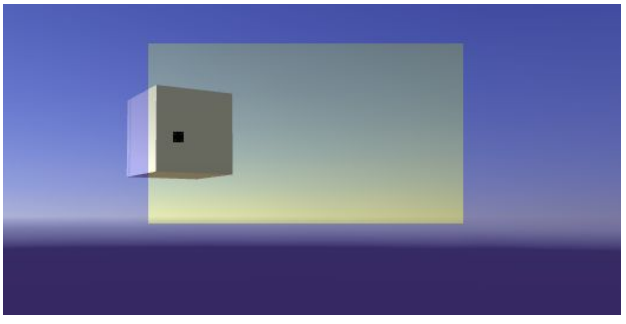

Figura 5.8: *Composer: Death Zone. Imatge de Creació pròpia.*

5.2.2 Transposer

El transposer és la segona funcionalitat més important de les càmeres bàsiques i esta encapsulat en una clase propia de la qual en te una referencia el BasicCamera. Simplificant molt el seu funcionament, el transposer es una versió del composer que en comptes de modificar la rotació per mantenir l'objecte en el focus de la càmera canvia la seva posició mantenint una distancia determinada (Offset).

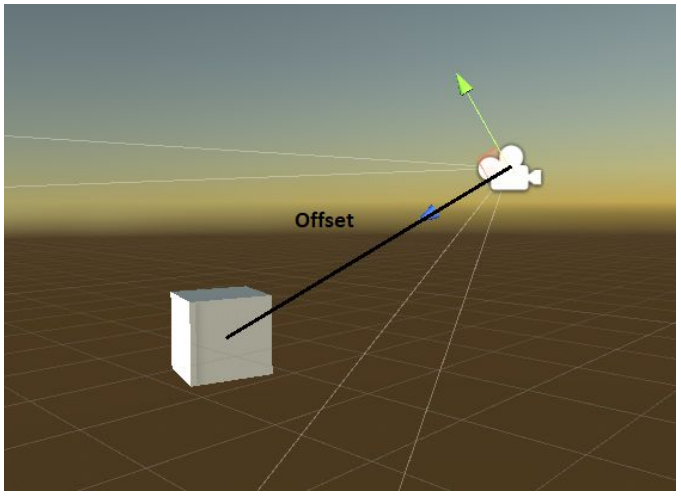

Figura 5.9: *Transposer: Offset. Imatge de Creació pròpia.*

El comportament que pot seguir també està desglossat en tres tipus:

- Seguiment de l'objecte segons un offset més moviment lliure de forma orbital.
- Seguiment de l'objecte segons un offset i utilitzant la rotació de l'objecte.
- Seguiment de l'objecte mantenint un offset i ignorant la rotació de l'objecte.

Cadascun d'aquest comportaments es descriuen a continuació.

1. **<u>Moviment Lliure.</u>**

El primer és un moviment lliure que permet a l'usuari orbitar a voluntat al voltant de l'objecte utilitzant els controladors HTC Vive. Per fer que el moviment sigui fluït i amb suavitat s'ha utilitzat una fórmula similar a la del posicionament de la càmera.

Quan no esta actiu el moviment lliure, activat per un input del controlador, la càmera segueix l'objecte amb un dels mètodes alternatius que s'expliquen més endavant. En el moment en que l'usuari decideix activar el moviment lliure, el programa agafa la posició actual del controlador. A mesura que el controlador s'aparta del punt de pivot guardat la velocitat de rotació al voltant de l'objecte s'incrementa proporcionalment.

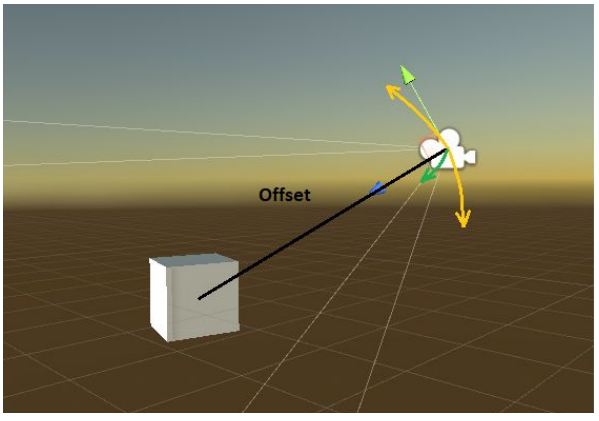

Figura 5.10: *Transposer: Moviment Lliure. Imatge de Creació pròpia.*

El vector format per la posició actual del controlador i el pívot representa la velocitat total de la rotació. Aquesta velocitat es separa en dues components, una per la velocitat en "y" i l'altre per la velocitat en el pla format per x i z. Les dues velocitats es multipliquen per una variable publica que pot modificar l'usuari i per la derivada del temps.

Un cop tenim les dues velocitats es calculen els quaternions que representen la rotació equivalent a aplicar en l'offset que dicta la posició de la càmera. Per calcular aquests dos quaternions utilitzarem el constructor de Quaternion proporcionat per Unity, Quaternion.AngleAxis(angle, eix), que genera un quaternio a partir d'un eix sobre el qual rotar i un angle en radians.

S'han de calcular dos quaternions, un per la rotació en x i un per la rotación en y. Per saber quins son els eixos sobre els que hem de rotar fem el cross product entre l'offset que uneix els dos objectes i el vector "Up" de l'objecte, vector "y" de la rotació local de l'objecte. Aquest primer cross product ens dona el nou vector fMRight. Si fem altre cop el cross product entre l'offset i el vector fMRight es pot calcular el vector fMUp.

```
Vector3 fMRight = Vector3.Cross(target.transform.up, fMOffset);
Vector3 fMUp = Vector3.Cross(fMOffset, fMRight);

fMOffset = Quaternion.AngleAxis(speed * speedX * Mathf.Deg2Rad, fMUp) * fMOffset;
fMOffset = Quaternion.AngleAxis(speed * speedY * Mathf.Deg2Rad, fMRight) * fMOffset;

camera.transform.position = target.transform.position + fMOffset;
```

Figura 5.11: *Transposer: Rotació Orbital . Imatge de Creació pròpia.*

Utilitzant el constructor AngleAxis amb el vector fMUp i l'angle a rotar en "x" i multiplicant-lo per l'offset rotem el vector en x l'angle desitjat. Repetim la operació amb l'eix fMRight i l'angle "y" i tindrem el nou vector Offset. Al aplicar aquesta relació s'obté un moviment molt suau i fluït al voltant del target.

2. Seguiment de la posició i rotació de l'objecte.

El segon tipus de moviment és un moviment que manté un offset i rota amb l'objecte. El seguiment de la rotació es fa calculant la funció "Quaternion.Lerp(Q0, Q1, t)" entre la rotació de l'objecte en el frame anterior, Q0, i l'actual, Q1. Aquesta rotació intermitja es calcula fent una interpolació lineal entre les dues rotacions en funció d'una variable de temps, t. Si es calcula la interpolació amb el temps sent igual a 0, la nova rotació sera la mateixa que la del frame anterior. Per contra si el valor del temps es igual a 1, aquesta sera la rotació actual.

Amb aquesta interpolació es produeix un delay entre el gir i el posicionament de la càmera a l'offset girat.

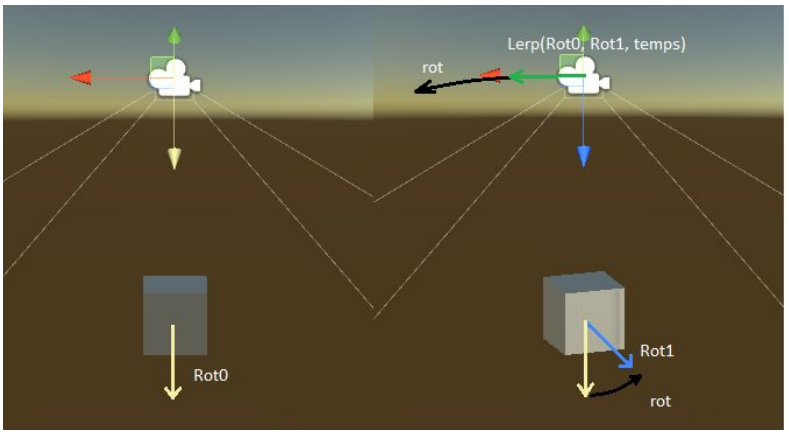

Figura 5.12: *Transposer: Rotar amb l'objecte. Imatge de Creació pròpia.*

3. Seguiment de la posició ignorant la rotació.

Si s'ignora la rotació, l'únic factor a tenir en comptes és la distància offset entre la càmera i l'objecte. Aquesta distancia sera constant per aquest cas en concret, per tant el vector offset no pateix cap canvi i pot passar a modificar la posició de la càmera per mantenir-se a la distancia correcte.

Un cop calculat el valor de Offset seguint qualsevol de les fórmules anteriors es sitúa la càmera a una distancia Offset de l'objecte i es centra en l'objecte amb la funció de càmera LookAt(posició de l'objecte).

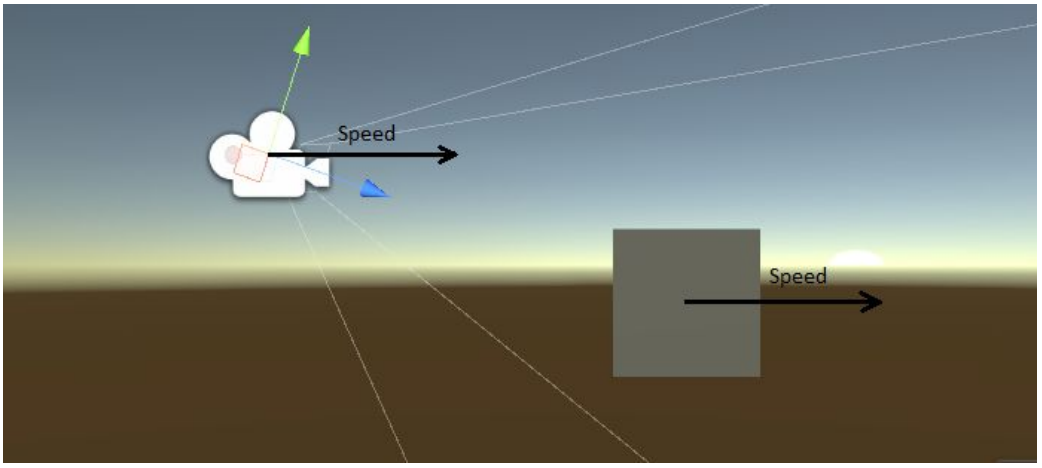

Figura 5.13: *Transposer: Fixed Offset. Imatge de Creació pròpia.*

5.3 Cinemachine Path

Cinemachine Path consisteix en un camí que indica la posició de la càmera en un instant en el temps.

5.3.1 Corbes de Bézier

L'aproximació al path de Cinemachine que es tracta en aquest projecte utilitza un conglomerat de corbes de Bézier que s'uneixen per crear el camí.

Les corbes de bézier són funcions polinòmiques de "n" graus de complexitat que a partir d'un punt inicial, un de final i un conjunt n-2 de punts de control creen una corba que començant al punt inicial estirigeix en direcció als diferents punts de control sense arribar mai a tocar-los.

La generalització de la fórmula per calcular un punt B(t) en una corba de bézier de grau n és la següent:

$$\mathbf{B}(t) = \sum_{i=0}^{n} \binom{n}{i} \mathbf{P}_i (1-t)^{n-i} t^i = \mathbf{P}_0 (1-t)^n + \binom{n}{1} \mathbf{P}_1 (1-t)^{n-1} t + \cdots + \mathbf{P}_n t^n \ , t \in [0,1].$$

Figura 5.14: *Generalització de la fórmula de Bézier. Imatge de Creació pròpia.*

Els valors $\binom{n}{i}$ corresponen als valors de la campana de pascal seguint la fórmula que és veu a continuació en funció del grau de l'equació polinomica.

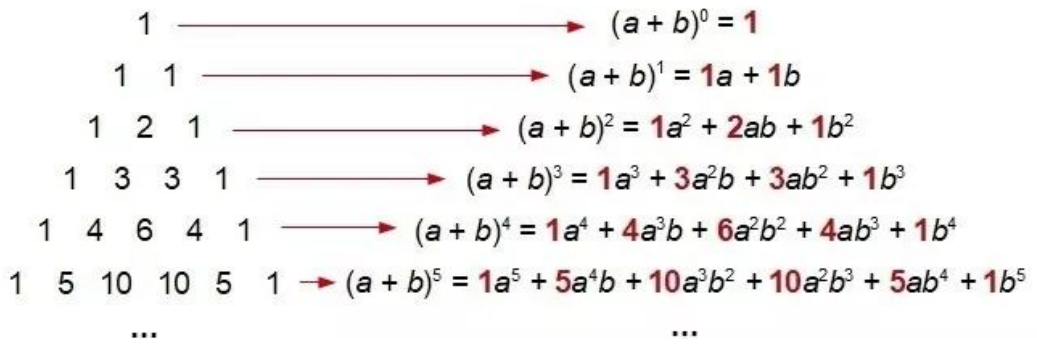

Figura 5.15: *Campana de Pascal. Extreta de Matemáticas Cercanas, última consulta 05/2019*

Utilitzant aquestes fórmules es pot calcular qualsevol punt de la corba de bézier, no obstant a major grau s'hagi d'evaluar la corba més complex sera el codi, comportant també un cost més elevat de computació i una baixada en el rendiment molt important.

El recorregut de les càmeres pot fluctuar constantment amb pujades (tangent positiva) i baixades (tangent negativa) de la seva posició. Si es té en compte que cada cop que la tangent d'una corba té un valor de 0 s'afegeix un grau a l'equació, és fàcil entendre que per representar un recorregut d'una càmera pot ser necessària una equació d'un grau molt elevat.

Un altre punt a tenir en compte és que les dades obtingudes del recorregut de la càmera són un conjunt de mostres de B(t), entre elles el P(i) quan i = 0 i el P(i) quan i = n, inici i final de la corba. Al no tenir els punts de control no es pot utilitzar la generalització de la corba de bézier per calcular la posició d'un punt B(t).

Una opció seria resoldre l'equació amb n graus de llibertat i calcular així tots els punts de control pero la variabilitat del valor "n" fa que sigui imposible de programar i quan major fos el valor de n més temps tardaria a calcular-ho.

La solució implementada a aquest problema és la divisió de la corba de grau 'n' a corbes cubiques de Bézier, és a dir corbes polinomiques de tercer grau. La conjunció d'aquestes corbes és un conglomerat de corbes de Bézier.

$$\mathbf{B}(t) = \mathbf{P}_0(1-t)^3 + 3\mathbf{P}_1 t(1-t)^2 + 3\mathbf{P}_2 t^2(1-t) + \mathbf{P}_3 t^3 \ , t \in [0,1].$$

Figura 5.16: *Fórmula: Corba Cúbica de Bézier. Imatge de Creació pròpia.*

Per crear aquest fragments del conglomerat s'ha de resoldre l'equació de tercer grau tinguent dos graus de llibertat (P1 i P2). Si s'avalua l'equació per dos valors de t diferents i es resol l'equació aïllant (P1 o P2) i substituint-la a l'altre s'obtenen els valors P1 i P2 per completar l'equació de la corba de Bézier i després calcular la resta de valors B(t). La qüestió està en triar bé quins valors de 't' a analitzar per obtenir el resultat més fidel possible de la corba. Aquests valors són t = ⅓ i t = ⅔ ja que divideixen la corba de forma equitativa en tres segments iguals dels quals ja se sap el principi i el final.

Així doncs un cop resolta l'equació ja es disposa de totes les dades necessàries del fragment i es passa a analitzar el següent fins a tenir tota la corba expressada en fragments de dos punts (inici P0 i final P3) i dos tangents (punts de control P1 i P2).

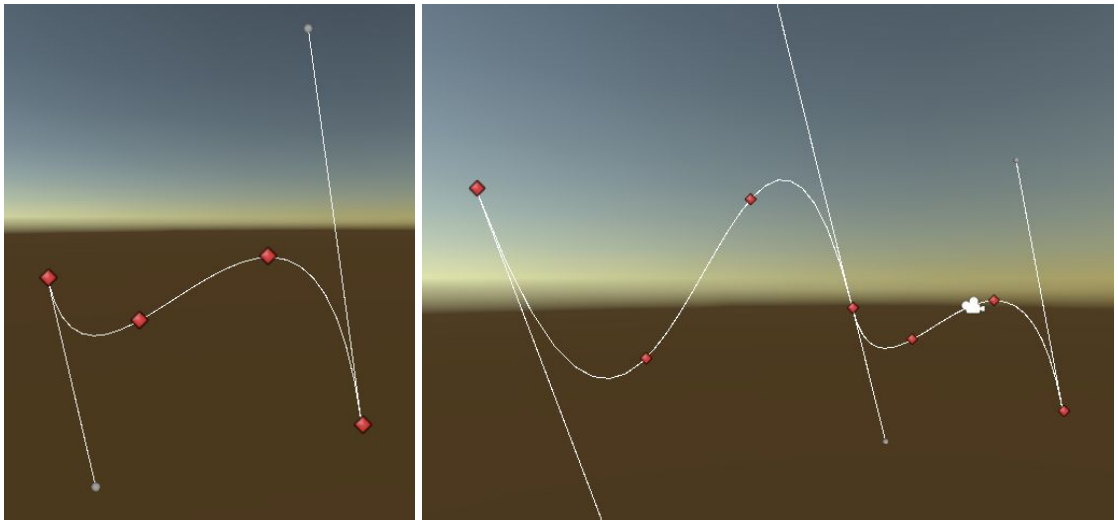

Figura 5.17 *i* ***5.18:*** *Corba Cúbica de Bézier i Conglomérat de Bézier. Imatges de creació pròpia.*

A la imatge de l'esquerra es pot veure el resultat de calcular una corba cúbica de Bézier on els punts vermells dels extrems son l'inici i el final i els punts intermitjos són B(t1) i B(t2) quan t1 = ⅓ i t2 = ⅔. Els segments acabats amb una esfera grisa representen els punts de control o tangents de la corba.

A la imatge de la dreta s'observa un conglomérat format per dos corbes cúbiques de Bézier.

5.4 Crear i Exportar Animation Clips

Els Animation Clips són un dels formats acceptats per Unity per representar modificacions de variables com són la posició, rotació o escalat d'objectes al llarg del temps.

5.4.1 Animation Curve

Els Animation Clips tenen una opció d'afegir Animation Curves, corbes que modifiquen el valor d'una variable segons el valor de la corba en l'instant de temps 't'. Dona la coincidencia que aquestes Animation Curves son funcions polinomiques on es pot afegir punts clau (KeyFrames) que indiquin el valor d'una variable en aquell instant del temps i la manera en que arriba la corba i en surt (tangent d'entrada i sortida).Precisament aquest valors son els que s'han extret del Cinemachine Path que també representen una funció polinòmica.

En l'apartat anterior s'obtenen el valor inicial i final de la corba a més de les tangents o punts de control. Si s'utilitzen els valors de les corbes de Bézier per definir els diferents keyframes, s'obté la corba que representa una aproximació a la funció polinòmica de grau 'n' original. Aquesta corba es pot associar a una variable com la posició 'x', 'y' o 'z' de la càmera.

El resultat del càlcul de les corbes de Bézier per a les posicions 'x', 'y' i 'z' de la càmera son tres Animation Curve diferents que permeten evaluar quina és la posició en un instant de temps de qualsevol de les tres variables.

Per crear aquestes corbes el primer que s'ha de fer és crear un array de Keyframes (Key[x]) amb una mida de (n+1) segments de Bézier. Començant per el primer punt, és defineix l'instant de temps corresponent, el valor de la variable en aquell punt (bezierPath[i, 0]), la tangent de sortida bezierPath[i,1] i la tangent d'entrada del segment de Bézier anterior (bezierPath[i-1, 2]). Es repeteix el procés fins haver definit tots els keyframes segons les dades i es crea la corba amb els Keyframes Key[].

```
// Curve for Position X ------------------------------------------------
float t = 0.0f;
int i = 0;
key[i].value = bezierPath[i, 0].x;
key[i].time = t;
key[i].outTangent = (bezierPath[i, 1].x - key[i].value) * 3;

for (i = 1; i < numBezierPath-1; i++)
{
    t += cLifespan[i];
    key[i].value = bezierPath[i, 0].x;
    key[i].time = t;
    key[i].inTangent = -(bezierPath[i-1, 2].x - key[i].value) * 3;
    key[i].outTangent = (bezierPath[i, 1].x - key[i].value) * 3;
}

i++;
t += cLifespan[i];
key[i].value = bezierPath[i, 3].x;
key[i].time = t;
key[i].inTangent = -(bezierPath[i, 2].x - key[i].value) * 3;

curves[0] = new AnimationCurve(key);
```

Figura 5.19: *Creació d'Animation Curve. Imatge de Creació pròpia.*

En el codi del projecte s'ha utilitzat un array de dues dimensions, la primera dimenció indica el número de corbes cúbiques de Bézier i la segona els 4 components de la corba {P0, T0, T1, P1} on P0 i P1 són el punt inicial i final i T0 i T1 són les tangents de sortida i d'entrada respectivament.

Per defecte les Animation Curves suavitzen les tangents d'entrada i sortida modificant la corba en si. Per evitar que això passi s'ha d'iterar tots els keyframes de la corba i ajustar el paràmetre TangentMode a 'Broken', acció que permet tangents d'entrada i sortida diferents.

5.4.2 Animation Clips

Des del document RecordPath, cada VR Camera podrà enregistrar el seu propi recorregut i generar les corbes de bézier pertinents en acabar de grabar el recorregut. Un cop enregistrat el recorregut aquest es podra exportar des d'un script extern, ExportPath, situat al VR Camera Manager. L'ExportPath s'encarrega de demanar quina és la càmera i recorregut a guardar, n'extreu les corbes d'animació, les afegeix a un AnimationClip i les associa a la variable de la posició o rotació corresponent amb la funció 'AnimationClip.SetCurve(...)'.

```
AnimationCurve[] curves;
if (path.HavePath())
{
    curves = path.RetrievePosAnimationCurve();

    clip.SetCurve("", typeof(Transform), "position.x", curves[0]);
    clip.SetCurve("", typeof(Transform), "position.y", curves[1]);
    clip.SetCurve("", typeof(Transform), "position.z", curves[2]);
}
```

Figura 5.20: *Creació d'Animation Clips. Imatge de Creació pròpia.*

Des del VR Camera Manager hi ha l'opció de probar el path resultant per saber si necessita alguna modificació ja sigui de la corba o de la duració dels diferents segments. Un cop verificat que el segment és el que es necessita els usuaris poden exportar l'Animation Clip. Aquesta exportació significa la generació d'un Asset AnimationClip que es guardara a la carpeta "Assets/" del projecte actual de Unity.

Un cop es té el clip amb totes les corbes que defineixen el camí que ha de recorrer la càmera hi ha l'opció de es crea un asset (AnimationClip) on guardar les dades.

```
if(clip)
{
    string name = "Assets/" + clipName + ".anim";
    AssetDatabase.CreateAsset(clip, name);
    AssetDatabase.SaveAssets();
}
```

Figura 5.21: *Guardar Animation Clips. Imatge de Creació pròpia.*

Els assets generats es poden importar a qualsevol projecte de Unity, modificar les diferents corbes dels AnimationClips i assignar l'animació a qualsevol objecte, sigui o no una càmera.

5.5: Editar les corbes de l'animació

Per donar més utilitat a l'eina, és indispensable implementar una funcionalitat per editar les corbes ja creades. Aquesta funcionalitat és vital per poder corregir errors o imprecisions resultants del càlcul de la corba o d'un moviment involuntari del dissenyador.

5.5.1 Visualització de la corba

El primer pas per poder editar la corba és visualitzar-la. S'utilitza la component Line Render, component disponible a Unity que crea un seguit de segments que uneix 'n' punts, per representar la línia en l'escena 3D i un seguit de models per representar els punts a modificar.

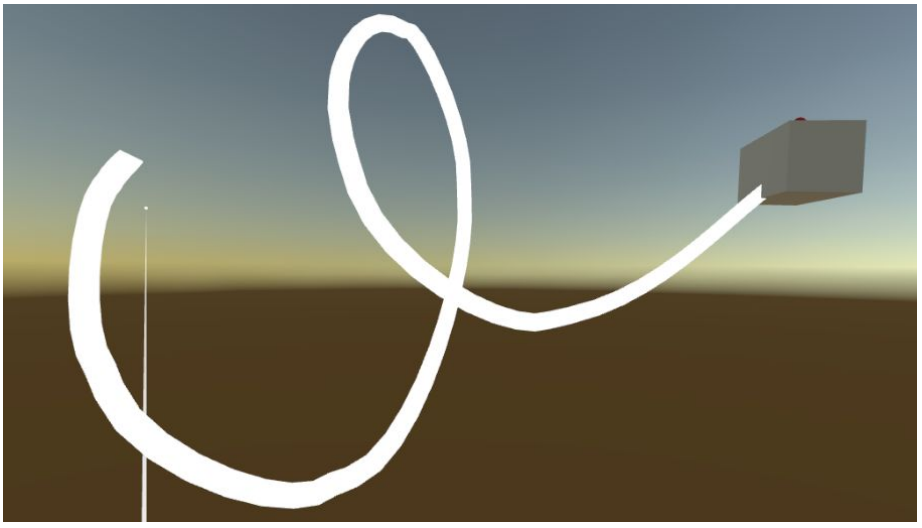

Figura 5.22: *Visualització Path. Imatge de Creació pròpia.*

Cal recordar que la corba esta formada per un conjunt de corbes de bézier i que aquests tenen un inici i un final que s'encadenen per formar l'Animation Curve. L'inici i el final de cada corba són els keyframes consecutius de l'animació, al modificar-ne el valor (posició) o les tangents (punts de control) es modifica el segment en si.

Si es mostres en pantalla tots els keyframes i tangents, tant d'entrada com de sortida seria confús per l'usuari, per tant, és limitarà la visualització al segment actual, i només és mostraran la posició inicial i final, la tangent de sortida del keyframe inicial i la d'entrada del keyframe final.

5.5.2 Identificació del Segment

Un cop presentada la corba en l'escena falta identificar quin segment es vol editar. La selecció del segment es farà incrementant una variable entera, 'i', que representarà l'índex del keyframe inicial del segment. El valor d'aquesta variable es modifica amb l'ajuda dels controladors de VR i a mesura que varia, els gizmos de transformació és faran visibles en els punts inicial i final del segment i les posicions de les tangents.

5.5.3 Edició dels valors

Com ja s'ha esmentat, cada keyframe té un valor, un temps i dues tangents, una d'entrada i una de sortida. Ignorant la variable 'temps', la modificació del 'valor', representa moure la posició, inicial o final, de l'objecte animat en l'instant de temps especificat en el keyframe. Una modificació en el valor de les tangents influeix en el "camí" que segueix l'objecte animat fins arribar al valor, posició, i com varia aquest valor fins arribar al següent keyframe.

Tenint en compte que el que interessa és modificar el segment comprès entre els keyframes inicial, k(i), i final, k(i + 1), s'haurà de tenir en compte el valor dels dos Keyframes, la tangent de sortida de k(i) i la d'entrada de k(i + 1).

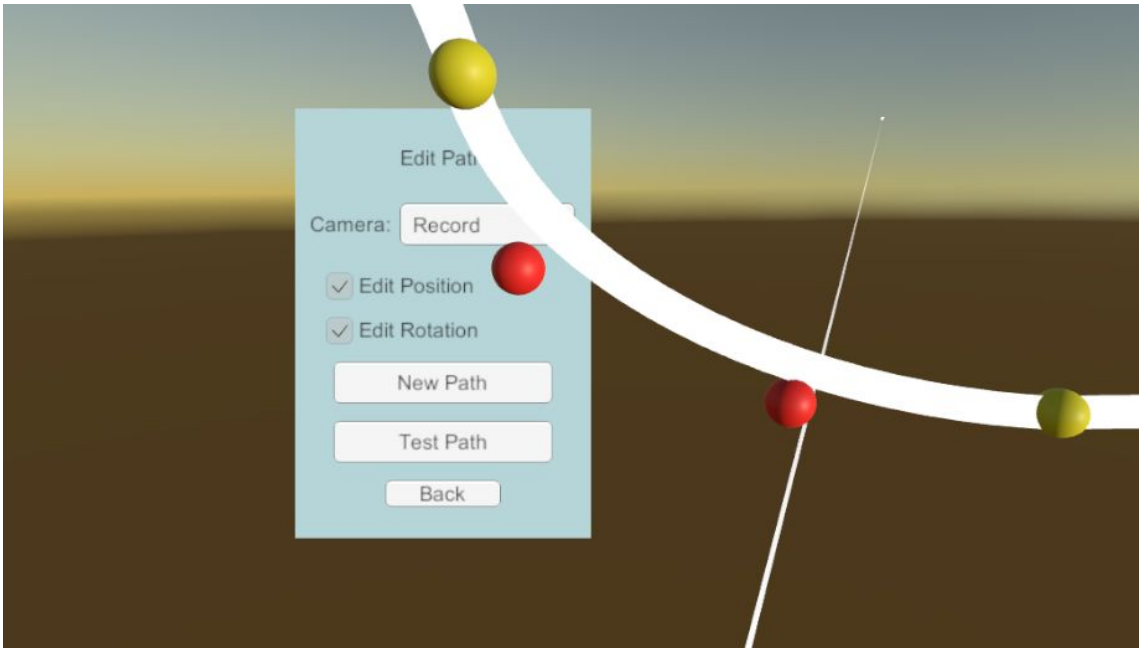

Figura 5.23: *Edició del Path. Imatge de Creació pròpia.*

És important recordar que en el moment de definir els keyframes a partir de la corba creada, s'ha passat els punts de l'escena al format utilitzat per Unity. Si es vol extreure o modificar la informació dels keyframes és necessari executar el procés invers per poder tenir les dades originals.

Les quatre variables originals a modificar són, en realitat, punts dins d'un espai 3D. Si es situen els diferents Gizmos de transformació amb colliders en els punts especificats, es poden utilitzar les col·lisions amb els controladors per seleccionar especificament quina de les variables modificar.

Un cop seleccionat quina variable modificar, s'utilitzen les fórmules utilitzades en el posicionament de càmeres estàtiques i és modifica el valor. En l'instant en què els valors són modificats, s'actualitzen els valors dels keyframes modificats i es calcularen de nou els punts introduïts al component Line Render.

5.6 Dolly Track

Com s'ha comentat en l'apartat 2. Estat de l'art, el dolly track és un path que a més de animar la posició de la càmera, guarda les modificacions de la rotació durant el recorregut. Aquesta rotació és independent dels keyframes utilitzats per definir la corba que modifica la posició, Animation Curves creats a partir de corbes de bézier.

Si bé es pot coincidir en els mateixos punts o keyframes també en poden aparèixer d'intermitjos en els diversos segments creats. Aquests nous keyframes definiran tres Animation Curves que s'afegiran al Animation Clip de la corba ja calculada i definiran la rotació de la càmera a mesura que avanci el temps.

Per seleccionar el punt on definir un keyframe s'utilitza el mateix sistema que per l'edició de corbes per seleccionar els keyframes propis del segment de la corba. També s'afegeix la possibilitat d'afegir un keyframe en mig del segment.

Aquest keyframe intermig pot ajudar a evitar rotacións no desitjades molt propies de l'animació on en comptes de rotar en un sentit rota en el sentit contrari al considerar que l'angle a recorrer és més petit. Per trobar aquest punt intermig on es posarà el keyframe es moura un GameObject avançant o retrocedint el temps i avaluant en quin punt esta l'objecte en l'instant desitjat. A la pràctica serà com un scroll d'un objecte a través de la corba.

Quan es tingui el keyframe en l'instant desitjat es passa a reposicionar tenint en compte només la rotació, utilitzant el mateix mètode que en el reposicionament.

A diferencia del path anterior, les tangents d'aquests keyframes que modifiquen la rotació utilitzaran la interpolació per defecte.

```
if(clip)
    clip.ClearCurves();
else clip = new AnimationClip();

RecordPath path = camera.GetComponent<RecordPath>();
if (path && path.HasPath())...
    // Set Position Curves

DollyTrack dolly = camera.GetComponent<DollyTrack>();
if (dolly && dolly.HasRotation())...
    // Set Rotation Curves

SaveAnimationClip();
```

Figura 5.24: *Exportar Dolly Track. Imatge de Creació pròpia.*

5.7 Interfície d'usuari

Per poder utilitzar totes les funcionalitats desenvolupades fins el moment, la última fase d'aquest projecte és el disseny i desenvolupament de la interfície d'usuari (UI). A l'hora de dissenyar el funcionament d'aquesta UI, cal tenir en compte dues seccions diferenciades. L'aspecte visual que tindrà la UI i com interacciona l'usuari amb aquesta.

5.7.1 Interfície Gràfica

L'aspecte visual de la interfície d'usuari constarà principalment de diferents menús i finestres integrades en l'escena 3D. Habitualment, gran part de les interfícies d'usuari es visualitzen directament en coordenades de pantalla (2D), és a dir es projecten sobre el Viewport.

El fet de tenir la UI en un espai 3D és important ja que l'usuari no disposa de teclat o ratolí per interaccionar amb la UI projectada sobre el Viewport. Com a tal, la UI propia de l'eina estarà dins de la escena. Per poder facilitar la seva visualització en tot moment, es posicionarà tots els elements respecte la càmera, visor de Realitat Virtual. Així en el moment en que l'usuari giri el cap, la UI també es mourà.

Entre els menús del prototip hi ha els següents:

- Menú Principal: enllaça amb la resta de menús principals.
- Menú de Creació de càmeres: permet crear càmeres amb valors per defecte de les diferents funcionalitats.
- Menú d'Exportació: s'encarrega de crear l'animació de la càmera seleccionada i exportar l'asset corresponent
- Menú d'Edició: permet activar o desactivar el mode edició.

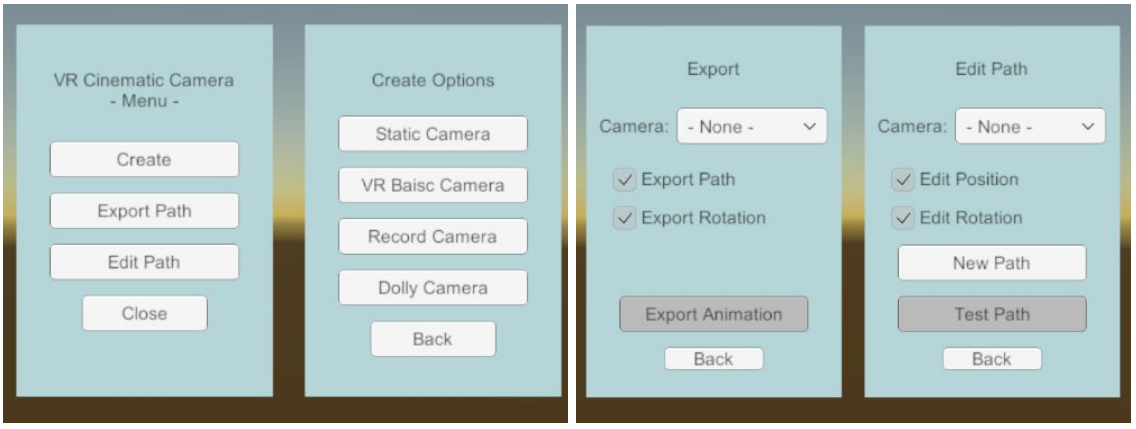

Figura 5.25 i 5.26: *Menú Principal i Create, Menú Export i Edit. Imatge de Creació pròpia.*

En obrir la UI, s'obre per defecte el menú principal, 1er menú de la figura 5.25, des d'aquest menú es pot accedir als altres pulsan els botons visibles, Create, Export Path i Edit Path.

A més dels menús principals, també hi ha finestres per configurar les variables pròpies dels scripts de les càmeres

- Configuració del BasicCamera: script que conté la funcionalitat del composer i el transposer.
- Configuració del RecordPath: script que defineixen com es registra el recorregut de la càmera
- Configuració del DollyTrack: script que defineix com es registra la rotació de les càmeras

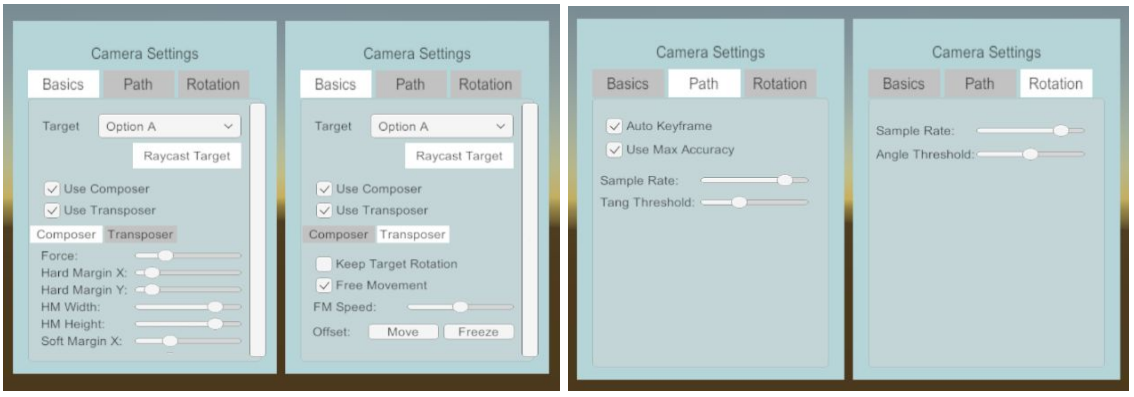

Figura 5.27 i 5.28: *Configuració Basics, Configuració Path i Rotation. Imatge de Creació pròpia.*

Respecte a les figures 5.27, 5.28, les pestanyes Basics, Path i Rotation configuren els scripts BasicCamera, RecordPath i DollyTrack respectivament. Les diferents pestanyes només es visualitzen si la càmera actual conté l'script corresponent.

I altres interficies d'usuari com són

- Dropdown: pestanya per canviar la càmera actual entre una llista de totes les càmeres creades
- Icona de gravació: informa a l'usuari de quan està gravant

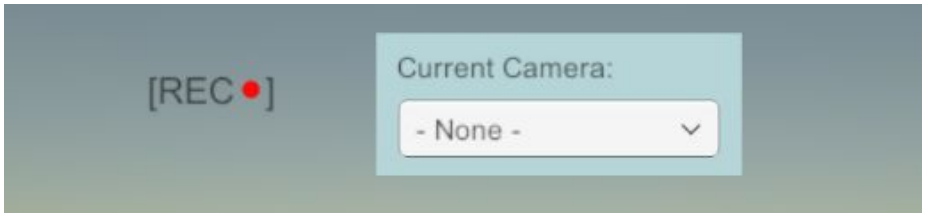

Figura 5.29: *Altres Interficies d'Usuari. Imatge de Creació pròpia.*

A més d'aquestes UI també hi ha la possibilitat d'activar una pantalla de suport que utilitza com a textura propia la textura on es renderitza l'escena des del punt de vista de la càmera actual de l'eina.

5.7.2 Interacció amb la UI

En interficies d'usuari 2D, l'usuaris disposa d'un punter (ratolí) i un teclat amb els quals poden navegar per la pantalla i interaccionar amb els diferents menús, buttons, etc.

Si ens centrem en l'ús del ratolí, la interacció es produeix al comparar les coordenades en pantalla del ratolí amb les de la UI. Es podria dir que es dispara un raig des de l'origen de la càmera passant per la posició del ratolí i s'activa la funcionalitat incorporada en el primer element de UI amb el qual xoca el raig.

La interacció amb la interfície 3D seguirà el mateix principi, aquest ratolí 3D serà molt similar a un punter laser que tindrà el seu origen al controlador dret. El funcionament d'aquest punter esta separat en dos scripts, un s'encarrega de la visualització del laser i l'altre de interaccionar amb la UI. En els dos scripts és necessari utilitzar un dels elements més utilitzats en el desenvolupament de videojocs, un raycast. El raycast és, com diu el nom, la creació d'un raig, aquest raig té un punt de sortida, una direcció i una llargada. Aquests rajos permeten detectar col·lisions amb diferents tipus d'objectes o elements.

Hi ha varis tipus de raycasts, i els utilitzats en els dos scripts comentats són fonamentalment diferents. En el cas de la visualització, el raycast és físic, ja que interessa saber amb quin element xoca el punter i finalitzar la visualització del laser en el punt on colisiona amb l'altre objecte.

Per contra, en la interacció es tracta d'un raycast gràfic, aquest raig es dispara des del controlador dret i torna el primer element de UI amb el que entra en contacte. Perquè es pugui crear aquest raig des del controlador, és necessari incorporar al gameobject del controlador un component càmera desactivat. El component càmera és el que dispara el raig en la seva direcció frontal i obté l'element de UI on està situat el cursor virtual. Un cop es té el cursor sobre l'element de UI només falta la interacció directa que ve donada per l'input dels controladors, el que equivaldra al clic esquerre del ratolí.

Si ve no tota la interacció es realitzarà amb el punter creat, la interacció directa amb els menús i botons serà majoritàriament amb l'ús d'aquest punter. Per la resta de UI i funcionalitats s'utilitzen els inputs dels controladors enllaçant-los amb diverses funcionalitats de forma més directe. Un exemple seria que al apretar un botó, s'obre el menú principal.

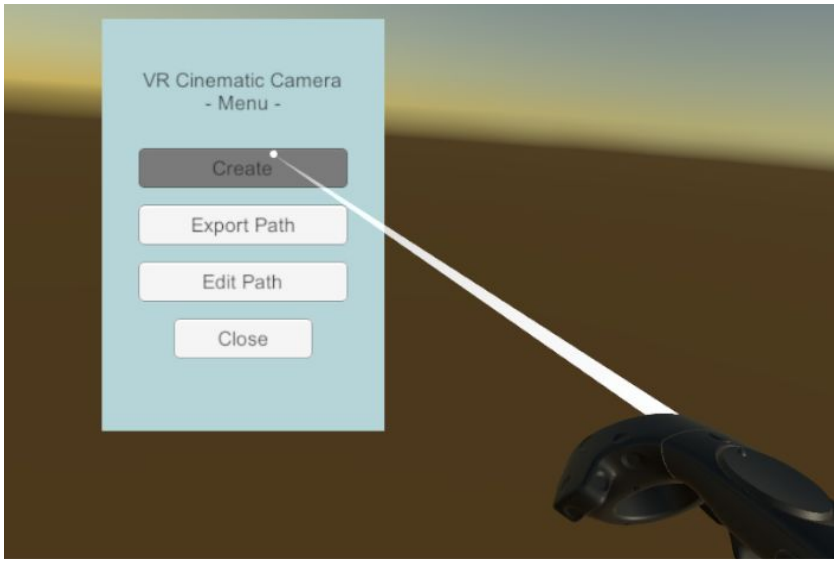

Figura 5.30: *Punter 3D. Imatge de Creació pròpia.*

6. Conclusions i treballs futurs

El contingut d'aquest treball està enfocat a obtenir coneixement sobre la utilització de les cutscenes com a eina de transmissió d'informació, l'anàlisi de les tècniques i eines utilitzades en la indústria per produir-les i la inclusió de la tecnologia de VR com a eina de treball.

Al llarg del desenvolupament del projecte s'han assolit varies métes. La primera fita assolida és l'adquisició d'un coneixement general del funcionament de les cutscenes amb un focus en les animacions de càmera en l'entorn de Unity. A partir d'aquest estudi podem entendre que les cutscenes són un recurs recurrent dels continguts multimedia per comunicar i transmetre informació als usuaris. Industries com la dels videojocs en fan un ús intensiu per tal de millorar l'experiència dels usuaris, alertant als jugadors del que els envolta, transmetent perills i oportunitats.

El mètode de producció de les cutscenes pot variar enormement en funció del públic objectiu i dels recursos tecnològics disponibles. Es pot dir que el sector dels videojocs és el que es veu més influenciat per les limitacions tecnològiques dels usuaris. Aquest fet es fa evident al veure les tendències en el mètode escollit per els desenvolupadors a l'hora de produir-les.

La desaparició dels Live-action és un cas exemplar de la influència de les capacitats tecnològiques dels equips dels usuaris en la producció de cutscenes. A finals del segle passat, els videojocs ocupaven molt poc espai en memoria i això donava la possibilitat de gravar seqüències amb actors reals per reproduir-les en moments determinats del videojoc. A mesura que passaven els anys, la capacitat i qualitat dels gràfics va augmentar exponencialment. Els videojocs van passar a tenir models més complexos i textures de més resolució. Aquest augment en la resolució tant de models com de textures va suposar un augment equivalent en la memoria ocupada i va reduir l'espai disponible per guardar Live-action cutscenes. Adaptan-se a les noves circumstàncies van prendre el relleu dels Live-Action el Pre-Rendered i en menor grau el Real-Time.

En l'actualitat, l'evolució tecnológica ha inclinat novament la balança, aquesta vegada en favor del Real-Time. Així les empreses del sector aposten cada cop més per el real-time, que, a pesar de no obtenir el mateix resultat en quant a gràfics, aporta altre beneficis, com un menor consum de memoria, costumització dels personatges i un major grau d'immersió dels jugadors.

Veient aquesta aposta per el Real-Time, s'han analitzat les opcions més rellevants per produir cutscenes dins de Unity, enfocant l'anàlisi en un dels components troncals de les cutscentes, les animacions de càmera.

La producció de animacions de càmera varia segons el mètode escollit per el desenvolupador. Hi ha tres opcions dins de Unity on es poden englobar la majoria d'animacions de càmera que s'hi reprodueixen. La primera és la de programmar els moviments de la càmera de forma manual via scripting. La segona és la d'utilitzar software de tercers per crear una animació i després importar-la a Unity. Per últim hi ha l'opció de utilizar plugins que ajudin a crear l'animació directament des de l'editor.

Considerant els dos primers mètodes com a més clàssics i d'un abast molt extens, han estat descartats com a matèria d'investigació. Per contra s'ha analitzat amb més cura la utilització de plugins dins de unity. Els plugins són una de les eines més influenciades pels avenços tecnològics i millores del motor. El fet de estar subjectes a les capacitats del motor fa que,

quan aquestes augmenten, l'eina o plugin es torni ràpidament obsolet. El curt temps de vida útil dels plugins i la constant evolució que pateixen els que sobreviuen, fa del desenvolupament de plugins un camp de treball obert a les noves tecnologies.

Observant els plugins orientats a la creació d'animacions de càmera, Cinemachine destaca com l'eina professional de preferencia per els usuaris. L'excel·lent resultat al que es pot arribar utilitzant algorismes com el Composer o el Transposer, és fruit de l'esforç realitzat per els desenvolupadors per simular un moviment de càmera natural més propi del cinema al que estem habituats. De fet, la similitud a una gravació propia del cinema fa que es percebi com a natural, millorant la immersió del jugador en l'entorn del videojoc.

Entre les altres funcionalitats de cinemachine també destaquen la opció de crear camins per on es desplaça la càmera al llarg del temps o en funció de la posició d'un objectiu. Aquests camins equivalents al recorregut del càmera es poden combinar amb el composer per crear una animació d'alta qualitat.

Prenen com a referencia les funcionalitats esmentades, un pas cap a la següent generació de plugins de animacions de càmera podria ser l'adaptació d'aquestes funcionalitats a una de les noves tecnologies emergents, la realitat virtual.

L'any 2017, quan la realitat virtual va iniciar un ascens continu en popularitat, hi va haver un primer intent fallit d'incorporar la realitat virtual en el procés de creació d'animacions de càmera. L'eina desenvolupada, VR Cameraman, explorava aquest nou concepte, això si, sense incorporar funcionalitats tan complexes com les propies de cinemachine.

Veien el potencial d'aquesta nova tecnologia i la possibilitat que suposa pels desenvolupadors el fet de disposar d'una nova eina, es va establir com a objectiu d'aquest treball l'estudi i desenvolupament d'un prototip que incorpores les funcionalitats destacades de Cinemachine en un entorn de realitat virtual.

Amb aquest objectiu en ment s'assoleix la segona fita del treball, la creació del prototip VR Cinematic Camera. En l'estudi de l'art, s'han investigat una serie de funcionalitats de Cinemachine així com de VRCameraman. De les funcionalitats estudiades, s'escull les més interessants per als desenvolupadors i que a la vegada poden tenir compatibilitat amb la realitat virtual. El resultat és la creació d'un prototip amb funcionalitats similars al composer, transposer, cinemachine path i dolly track adaptats a Realitat Virtual.

La "Similitud" amb les funcionalitats del plugin estudiat, fa referència a la incapacitat d'accedir al codi privat dels plugins a replicar. El fet de no tenir cap codi de referencia implica sotmetre's a una dura fase d'investigació on amb suposicions, més o menys ben fundades, s'aconsegueix imitar els algorismes objectius amb un resultat satisfactori. Si ve l'únic cas en el que s'ha intentat replicar fidelment l'original és el del composer, no es pot negar que la resta de funcionalitats han partit de la base conceptual ja establerta per Cinemachine.

El desenvolupament del prototip es dur a terme centrant-se en el disseny i producció de funcionalitats, una per una, partint d'un codi base al qual se li afegeixen les diferents peces. El codi base consta d'un gestor principal o manager que manté les funcionalitats troncals i gestiona les diferents càmeres creades per el prototip que hi hagi a l'escena. A partir d'aquest punt, l'ordre d'implementació es va estipular per poder testejar des de la primera fase i així entrar en un bucle de disseny, producció i testeig.

Després de passar una fase de conceptualització, es defineix una estructura piramidal amb un manager a la part superior que s'encarrega de gestionar les funcionalitats principals i les càmeres. Dins de les càmeres s'amaga un dels scripts més crucials que incorpora funcionalitats bàsiques com el posicionament de les càmeres en l'escena i funcionalitats avançades com són una aproximació al composer i al transposer adaptats per VR.

Aquí és on apareix el primer progress significatiu. A partir de la informació esmentada en l'estat de l'art sobre el composer i el transposer s'aconsegueix fer una aproximació fidel al seu comportament. En el cas del transposer s'inclou un moviment orbital mentre segueix a l'objectiu controlat a voluntat per l'usuari.

En aquest punt pren el relleu la creació d'un recorregut de la càmera que sigui compatible amb el comportament base de les càmeres. Per la creació d'aquest recorregut, s'ha considerat l'ús de funcions polinòmiques, corbes de bézier, per representar de forma senzilla els recorreguts. Al combinant corbes de bézier de tercer grau es pot crear un conglomerat de segments que formen un camí. cada segment està format per dos punts inici i final i uns punts de control. Amb aquests quatre punts guardats, hi ha la possibilitat de crear i editar AnimationClips, format utilitzat per Unity per representar variacións de varialbes al llarg del temps, en el nostre cas posició i rotació. Repetint un procés semblant per a la rotació de la càmera, conclueix el desenvolupament de funcionalitats principals.

Després de produir una interfície d'usuari que permet una interacció correcte amb l'eina es dóna per acabat el prototip. Si ve es considera que s'ha assolit un èxit relatiu en la creació del prototip, també queda clar que per llançar al mercat aquesta eina és necessària una fase extra per polir i millorar-ne el seu ús.

Revisió dels Objectius:

En quant als objectiu especificats a l'inici del projecte, a continuació és fa un desglossat analitzant el resultat.

1. **Creació d'un prototip per crear i editar recorreguts de càmeres en entorns 3D per al motor Unity.**

   El primer objectiu s'ha assolit, al crear una eina amb les funcionalitats desitjades adaptades a la realitat virtual. El prototip permet crear i editar recorreguts de càmeres dins de Unity en temps d'execució. També és veritat que el resultat del projecte no deixa de ser un prototip i que per tal de representar una solució real al problema plantejat, és necessari passar a una fase de polit, millora de interfície d'usuari i eliminació de bugs.

2. **Implementació de la realitat virtual com a eina per controlar les càmeres i el seu recorregut en l'etapa de rodatge.**

   Tant la interacció com les funcionalitats desenvolupades, a excepció del Composer i una part del Transposer, requereixen de l'ús de la realitat virtual per poder funcionar. Per tant es pot dir que el prototip deixa de ser funcional si no s'utilitza un equip de Realitat Virtual. Conseqüentment, l'objectiu d'implementar l'eina en Realitat Virtual està assolit.

3. **Implementació de funcionalitats avançades de càmera que siguin compatibles amb la Timeline de Unity.**

   En una primera fase del desenvolupament, hi havia la intenció de mantenir una execució entrellaçada del prototip i de la Timeline. Degut a les dificultats tècniques que suposa mantenir aquesta relació envers els beneficis, es va decidir mantenir separats la TimeLine de la creació de recorreguts. Un cop els recorreguts s'han enregistrats és duen a terme les operacions necessàries per crear un Asset compatible amb la Timeline que en aquest cas és l'Animation Clip i les Corbes d'animació. Com a tal, és possible incorporar qualsevol animació creada amb el prototip a la timeline, assolint també aquest objectiu.

4. **Proporcionar eines de suport per millorar la usabilitat per part dels dissenyadors que utilitzin l'eina.**

   L'eina de suport més important incorporada és la d'una pantalla on es pot veure l'escena pintada des de la càmera activa que s'està utilitzant en el moment. Aquesta eina és vital ja que des de la realitat virtual l'usuari només veu la escena des del punt de vista de les ulleres, i seria impossible aconseguir una animació adecuada sense veure que es el que s'està gravant.

5. **Rodatge d'una Cutscene senzilla que mostri les capacitats de l'eina.**

   La finalitat d'aquesta cutscene és la de crear una demo del prototip a presentar al tribunal. Precisament per aquest motiu en el seu moment es va definir com a objectiu extra, el rodatge del cual tindrà lloc els dies posteriors a l'entrega d'aquest document. En el cas que el projecte és publiqui a la Asset Store de Unity, el video estarà disponible en la descripció.

Treballs futurs:

La realització d'aquest treball ha suposat la primer pas en la creació d'un plugin funcional. En el mesos posteriors a l'entrega del document el prototip entrarà en una fase de millora de funcionalitats, interfície i interacció amb l'objectiu d'assolir la qualitat necessària per publicar-lo a la Asset Store de Unity amb el nom de "VR Cinematic Camera".

7. Webgrafia

Agile Methodologies for Software [en línia][consultat el dia 25/06/2019]
Enllaç: https://resources.collab.net/agile-101/agile-methodologies

Unity [en línia][consultat el dia 25/06/2019]
Enllaç: https://unity.com/

Unity Manual: Timeline [en línia][consultat el dia 25/06/2019]
Enllaç: https://docs.unity3d.com/Manual/TimelineOverview.html

Unity Tutorials: Animation-Timeline [en línia][consultat el dia 25/06/2019]
Enllaç:https://unity3d.com/es/learn/tutorials/topics/animation/using-timeline-overview?playlist=17099

Unity Tutorials: Animation-Cinemachine [en línia][consultat el dia 25/06/2019]
Enllaç:https://unity3d.com/es/learn/tutorials/topics/animation/using-cinemachine-getting-started

Unity Cinematics VR [en línia][consultat el dia 25/06/2019]
Enllaç: https://unity.com/es/solutions/cinematicvr

Cinemachine Imagery [en línia][consultat el dia 25/06/2019]
Enllaç: https://www.cinemachineimagery.com/

Cinemachine Documentation [en línia][consultat el dia 25/06/2019]
Enllaç: https://www.cinemachineimagery.com/base-rig/

Cinemachine Module Example [en línia][consultat el dia 25/06/2019]
Enllaç: https://www.cinemachineimagery.com/examples/

Geometria Dinàmica: Corbes de Bezier [en línia][consultat el dia 25/06/2019]
Enllaç: http://www.geometriadinamica.cl/2010/12/curvas-de-bezier/

A Primer on Bézier Curves [en línia][consultat el dia 25/06/2019]
Enllaç: https://pomax.github.io/bezierinfo/

Curva de Bézier - Wikipedia [en línia][consultat el dia 25/06/2019]
Enllaç: https://es.wikipedia.org/wiki/Curva_de_B%C3%A9zier

Unity Manual: Animation Clips [en línia][consultat el dia 25/06/2019]
Enllaç: https://docs.unity3d.com/es/current/Manual/AnimationClips.html

Unity Manual: Using Animation Curves [en línia][consultat el dia 25/06/2019]
Enllaç: https://docs.unity3d.com/es/current/Manual/animeditor-AnimationCurves.html

Unity Documentation: Scripting Api. CreateAsset [en línia][consultat el dia 25/06/2019]
Enllaç: https://docs.unity3d.com/ScriptReference/AssetDatabase.CreateAsset.html

Unity Documentation: Scripting Api. SaveAssets [en línia][consultat el dia 25/06/2019]
Enllaç: https://docs.unity3d.com/ScriptReference/EditorApplication.SaveAssets.html

Trello [en línia][consultat el dia 25/06/2019]
Enllaç: https://trello.com/

Instagantt [en línia][consultat el dia 25/06/2019]
Enllaç: https://instagantt.com/r